Proyecto Fin de Carrera

DISEÑO Y DESARROLLO DE UN SISTEMA DE DIAGNÓSTICO DE VEHÍCULOS BASADO EN MÓDULOS EMPOTRADOS DE BAJO COSTE Y EN DISPOSITIVOS MÓVILES INTELIGENTES

Alumno: **Juan Antonio Ruiz Mula**

Director de proyecto: **D. Juan Antonio López Riquelme**

Dña. Nieves Pavón Pulido

“Señor, si no tienes cubo, y el pozo es hondo,
¿de dónde sacas agua viva?”[11]
Jesús le contestó:
“el que beba del agua que yo le daré
nunca más tendrá sed […]”[14]
(Juan 4)

Al director de este PFC
por su paciencia
con este Ing. I.T. Mecánico

A los integrantes de los grupos
“C. Industriales” y “Fier. Trav.” por los
buenos momentos, en este oscuro sueño

Índice

Capítulo 1

Introducción

1.1 Introducción

La actual revolución tecnológica pone al alcance del usuario herramientas, que unidas al creciente interés por el estado de sus vehículos, permiten realizar diagnósticos de averías y medidas de parámetros en tiempo real e in situ con el simple uso de un Smartphone, *Tablet*, Pc, etc.

La electrónica ha complementado los sistemas clásicos de funcionamiento puramente mecánicos, hasta el punto de desplazarlos o incluso sustituirlos en algunos casos, optimizando el consumo de combustible, creando sistemas más flexibles y suaves, mejorando el confort, prediciendo averías, etc. Para la predicción de averías los automóviles cuentan con un puerto de entrada y salida de información, denominado OBD, que conecta con la Unidad de Mando (ECU) encargada del control y recopilación de todos los parámetros del vehículo.

Por otro lado, en la actualidad existe la posibilidad de crear aplicaciones software para dispositivos móviles inteligentes que funcionan gracias a sistemas operativos, tales como, Android e iOS, entre otros. Estos dispositivos móviles pueden ser utilizados para realizar muchas tareas, ya que incluyen una pantalla táctil que sirve tanto de interfaz como de teclado de control, que junto con los sistemas de conectividad (WiFi, Bluetooth, 4G), almacenamiento y procesamiento. Además, el desarrollador del sistema operativo suele ofrecer, tanto un IDE para el desarrollo de la denominada APP, como un kit de desarrolla software que facilita el uso del hardware del dispositivo móvil para las tareas necesarias.

1.2 Objetivos

El presente proyecto tiene como objetivo diseñar y desarrollar un sistema de diagnóstico de vehículos basado en módulos empotrados de bajo coste y en dispositivos móviles inteligentes, que además permita la lectura de parámetros y el borrado de averías de la Unidad de Mando (ECU). Para cumplir con el objetivo anterior se llevarán a cabo los siguientes sub-objetivos:

- Estudio y selección de los diferentes elementos que componen las arquitecturas hardware y software del sistema.
- Diseño y desarrollo da la arquitectura hardware del dispositivo de diagnóstico inalámbrico y con conexión con la interfaz OBD de la ECU.
- Diseño y desarrollo de la arquitectura software del dispositivo de diagnótico inalámbrico OBD.
- Diseño de una App para dispositivos móviles inteligentes Android, que permita interactuar con el dispositivo para realizar diagnósticos en vehículos equipados con unidad de control con interfaz OBD.
- Realización de pruebas y puesta en funcionamiento del sistema.

Capítulo 2

Estado del Arte

2.1 Introducción

Como se comentó en el primer capítulo, el objetivo principal es el diseño y desarrollo de un sistema de diagnóstico de vehículos basado en módulos empotrados de bajo coste y en dispositivos móviles inteligentes.

En el diseño y desarrollo del sistema de diagnóstico se pueden emplear los siguientes elementos: dispositivos empotrados de hardware y software libre, sistemas de diagnóstico de vehículos, sistemas operativos para dispositivos módulos inteligentes, y sistemas de comunicación.

En el presente capítulo se realizará un estado de la técnica de algunas alternativas posibles para los módulos mencionados. Del mismo modo, para diseñar y desarrollar la App móvil necesaria se analizarán los sistemas operativos para Smartphone más relevantes.

2.2 Dispositivos empotrados

Los sistemas empotrados open-source y de hardware libre, o sistemas embebidos, son sistemas de computación diseñados para realizar algunas funciones de sistemas de computación en tiempo real. A diferencia de los ordenadores de propósito general, que

están diseñados para cubrir un amplio rango de necesidades, los sistemas embebidos se diseñan para cubrir necesidades específicas.

Los sistemas empotrados tienen las siguientes características:

- Están constituidos por un microprocesador, microcontrolador, DSP, etc. Es decir, la CPU o unidad que aporta capacidad de cómputo al sistema, generalmente de tipo ARM cuando se trata de microprocesador.
- La comunicación adquiere gran importancia en los sistemas embebidos. Es normal que el sistema pueda comunicarse mediante interfaces estándar cableadas o inalámbricas.
- El subsistema de presentación habitual suele ser una pantalla gráfica, táctil, LCD, etc.
- Se denominan actuadores a los posibles elementos electrónicos que el sistema se encarga de controlar.
- El módulo de E/S analógicas y digitales suele emplearse para digitalizar señales analógicas procedentes de sensores, activar diodos LED, reconocer el estado abierto cerrado de un conmutador o pulsador, etc.
- El módulo de reloj es el encargado de generar las diferentes señales de reloj a partir de un único oscilador principal. En este módulo son importantes los aspectos de: frecuencia necesaria, estabilidad necesaria y consumo de corriente requerido.
- El módulo de energía genera diferentes tensiones e intensidades necesarias para alimentar los circuitos del sistema. Usualmente, trabaja con un rango de posibles tensiones de entrada que mediante conversores ac/dc o dc/dc, obtienen diferentes tensiones necesarias para alimentar diversos componentes activos del circuito.
- Los conversores ac/dc y dc/dc, otros módulos típicos, filtros, circuitos integrados supervisores de alimentación, etc.

2.2.1 Arduino

Arduino (ver Ilustración 2.1) es una plataforma de hardware libre, basada en una placa con un microcontrolador y un entorno de desarrollo, diseñada para facilitar el uso de la electrónica en proyectos multidisciplinares.

El hardware consiste en una placa con un microcontrolador Atmel AVR, siendo los más usados el Atmega168, el Atmega328, el Atmega1280, el ATmega8 y el ATmega32u4. Todos se caracterizan por su sencillez y bajo coste, permitiendo el desarrollo de múltiples diseños. El software consiste en un entorno de desarrollo que implementa el lenguaje de programación *Processing/Wiring* y un bootloader [1].

Desde octubre de 2012, Arduino también emplea microcontroladores CortexM3 de ARM de 32 bits y AVR de 8 bits más económicos. ARM y AVR no son plataformas compatibles a nivel binario, pero se pueden programar con el mismo IDE de Arduino para que compilen en ambas plataformas. Los microcontroladores CortexM3 usan 3,3 V, a diferencia del resto de las placas con AVR, que generalmente usan 5 V. Arduino se puede utilizar para desarrollar objetos interactivos autónomos, o puede ser conectado a software tal como Adobe Flash, Processing, Max/MSP, Pure Data). Las placas se pueden montar a mano o adquirirse. El entorno de desarrollo integrado libre se puede descargar gratuitamente.

Arduino puede tomar información del entorno a través de sus entradas y controlar luces, motores y otros actuadores. El microcontrolador en la placa Arduino se programa mediante el lenguaje de programación Arduino (basado en *Wiring*) y el entorno de desarrollo Arduino (basado en *Processing*). Los proyectos hechos con Arduino pueden ejecutarse sin necesidad de conectar a un computador.

Ilustración 2.1. ***Tarjeta de desarrollo Arduino UNO***

Arduino permite conectar diferentes extensiones o *shields* sobre la placa principal, aportando funcionalidades extra al dispositivo sin perder además las conexiones originales (salvo algunas excepciones).

2.2.2 Texas Instruments Lauchpad

Texas Instruments, más conocida en la industria electrónica como TI, es una empresa norteamericana con sede en Dallas (Texas, EE. UU) que desarrolla y comercializa semiconductores y tecnología para ordenadores. TI es el tercer mayor fabricante de semiconductores del mundo.

Las placas de desarrollo de la serie LaunchPad (ver Ilustración 2.2) son muy económicas (alrededor de 10 $), están basadas en micro-controladores de ultra bajo consumo de la serie MSP430, incorporan la herramienta de programación y depuración del código sobre la misma placa, e incluyen dos conectores de expansión de 20 pines para la conexión de las denominadas Booster-Pack. Este término hace referencia a placas específicas que dotan de funcionalidad extra a la placa base y son similares a los shiels para Arduino [2].

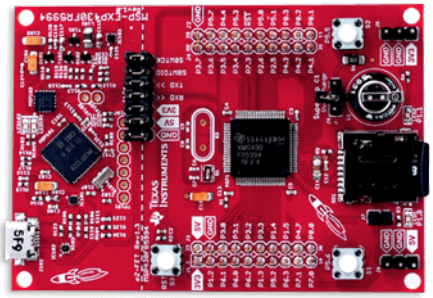


Ilustración 2.2. ***Placa de desarrollo MSP-EXP430FR5994 de la serie Launchpad de TI***

2.3 Sistemas de diagnóstico de vehículos

2.3.1 OBD II UART Hookup Guide

La tarjeta OBD II Uart Hookup Guide (ver Ilustración 2.3) es una placa de desarrollo que permite de una manera sencilla dotar a un sistema de interacción con unidades de control de vehículos. On-Board Diagnostics de Second Generation (OBD-II) consiste en un conjunto de normas para la implementación de un sistema, que permite controlar los parámetros de los vehículos mediante un dispositivo electrónico. Se introdujo por primera vez en los Estados Unidos en 1994, y se convirtió en un requisito para todos los vehículos estadounidenses de 1996, en adelante. Otros países, como Canadá, partes de la Unión Europea, Japón, Australia y Brasil adoptaron una legislación similar. La mayoría de la flota de vehículos actuales utiliza OBD-II o derivados.

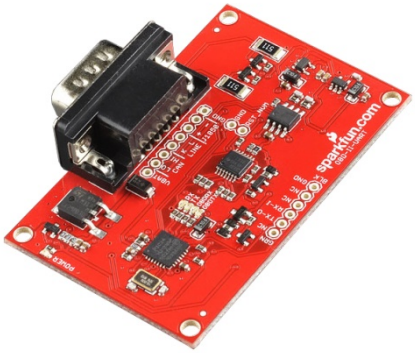


Ilustración 2.3. ***Sistema OBD II UART Hookup Guide***

OBD-II requiere que cada vehículo compatible esté equipado con un conector de diagnóstico estándar (DLC) y describa una forma normalizada de comunicación con el ordenador de a bordo del vehículo, también conocida como ECU (*Electronic Control Unit*). De esta unidad se puede obtener una gran cantidad de información mediante el bus OBD, incluida la avería indicada por la luz (chivato) de mal funcionamiento (MIL), los códigos de diagnóstico de problemas (DTCs), la información de inspección y mantenimiento (I/M), cientos de parámetros en tiempo real, etc.

El ELM327, que es el elemento principal de esta placa de desarrollo, es un microcontrolador programado y producido por ELM Electronics para traducir la interfaz de diagnóstico a bordo (OBD) que se encuentra en la mayoría de los coches modernos a comandos más sencillos (AT). El protocolo de comandos ELM327 es uno de los estándares de interfaz PC a OBD más populares y también es implementado por otros proveedores. A nivel de implementación es muy habitual encontrar que el dispositivo ELM327 está basado en el microcontrolador PIC18F2480 de Microchip Technology [3].

2.3.2 ELM327 OBD-II

El *ELM327 OBD-II* (ver Ilustración 2.4) es compatible con gran número de vehículos (ver Ilustración 2.5) que emplean el protocolo OBD-II, aunque sólo permite acceder a Unidades de Mando de vehículos muy concretas y en general únicamente se pueden leer los códigos de averías que reporta el motor, es decir, tiene restringido el acceso a datos de Airbag, ABS, Climatizador, etc. En algunos modelos ofrece más datos en tiempo real que en otros. Los códigos de averías obtenidos por el dispositivo son enviados como un código compuesto de letras y números [4].

Ilustración 2.4. ***Sistema comercial ELM327 OBD-II***

Existen diferentes formas de conexión para el ELM327 OBD-II, como USB, Bluetooth, WiFi, compatibles para Smartphone, PC, etc.

Reference Compatible vehicles :

Benz 2001/02/03 W203 C200K (ISO).
BMW X5 2002 (ISO) -3, -5/X5/Z3/Z4/Z8/740I/740L/750IL, (ISO1941) Chevrolet Aveo (ISO).
Citroen C3 2004(ISO).
Daihatsu YRV 2005(ISO).
Daihatsu YRV Turbo 2006(ISO).
Fiat Tipo 2002(ISO).
Ford New ocus 2006(CAN).
Honda Accord 2006 JDM(ISO).
Honda Accord Euro R 2004 JDM(ISO).
Honda City GD8(ISO).
Honda Civic ES 1.6A(ISO).
Honda Integra Type R DC5 JDM(ISO).
Honda Jazz 1.4M(ISO).
Hyundai Accent 2004(ISO).
Hyundai Getz 1.3A 2004(ISO).
Hyundai Getz 1.6M 2004(ISO).
Hyundai Matrix 2004(ISO).
Hyundai Santa Fe 2.4A(ISO).
Hyundai Sonata GLS(ISO).
Hyundai Sonata 2005(ISO).
Mitsubishi Lancer Evolution IX 2006(ISO).
Mitsubishi Pajero 3.5 V6 2006(ISO).
Nissan Sunny B14(ISO).
Nissan Sunny VIP FB15(ISO) .
Nissan X-Trail 2.0L 2004(ISO).
Nissan X-Trail 2.5L 2004(ISO).
Peugeot 206 2001 Only can show RPM.
Peugeot 206 (ISO).
Peugeot 407 (ISO).
Renault Megane II 2005(ISO).
Renault Kangoo 2006(ISO).
Kia Carens 2005(ISO).
Kia Magentis 2005(ISO).
Kia Rio LS (ISO).
Kia Sorento SUV 2005(ISO).
Lexus es300 Mazda 2(CAN).
Mazda 5 (CAN).
Mazda 6 (CAN).
Mazda Premacy 2001(ISO).
Mazda RX8 (CAN).
Mitsubishi Colt 2003 (ISO).
Mitsubishi Colt Plus 2005(ISO).
Subaru Impreza 1.6TS 2002(ISO).
Subaru Impreza WRX 2005 (ISO).
Nissan Sunny (N16).
Nissan Cefiro (Teana)2005.
Nissan sentra 1.8 2000/2005.
Nissan Cefiro (Teana)2005.
Proton Perdana V6.
Subaru Legacy GT 2005.
Peugeot 206 2000.
Suzuki Jimny 1.3 2000.
Toyota Camry 2.0A 2004 (ISO)
Toyota Camry 2.4A 2005 (ISO)
Toyota Corolla Altis 1.6A 2005 (ISO)
Toyota Corolla GLXi G8 1999 (ISO)
Toyota Corolla G9 2002 (ISO)
Toyota Vios 2004 (ISO)

Ilustración 2.5. ***Cuadro de vehículos compatibles con el ELM327 OBD-II***

2.3.3 Protocolo SAE para OBD-II

SAE International, inicialmente establecida como la Sociedad de Ingenieros Automotrices, es una asociación de profesionales y organización de estándares estadounidenses, activa a nivel mundial para profesionales de la ingeniería en diversas industrias. Está orientada, principalmente, a las industrias del transporte, más concretamente automoción, aeroespacial y vehículos comerciales.

SAE ha publicado más de 1.600 normas técnicas y prácticas recomendadas para automóviles de pasajeros y otros vehículos de carretera. También proporcionan referencias de la industria para la medición de la potencia del motor, clasificación de aceite de motor, tamaño de herramienta y tornillos, y conectores y códigos de diagnóstico a bordo. SAE también publica normas y prácticas recomendadas para luces, frenos, fluidos de transmisión automática, redes de comunicación, sistemas de carga de vehículos eléctricos, ergonomía de vehículos y más aspectos del diseño, la construcción, el rendimiento y la durabilidad del vehículo.

En relación a normas publicadas y relacionadas con OBD-II se pueden destacar las siguientes:

- J1962 - Define el conector físico para la interfaz OBD-II.
- J1850 - Define un protocolo de datos en serie. Hay dos variantes: 10,4 kbit/s (de un solo hilo, VPW) y 41,6 kbit/s (dos hilos, PWM). Principalmente, utilizado por los fabricantes estadounidenses, también conocido como PCI (Chrysler, 10,4 kbit/s), Clase 2 (GM, 10,4 kbit/s) y SCP (Ford, 41,6 kbit/s).
- J1978 - Define estándares de operación mínimos para las herramientas de exploración OBD-II.
- J1979 - Define los estándares para los modos de prueba de diagnóstico.
- J2012 - Define códigos y definiciones de problemas de estándares.
- J2178-1 - Define estándares para formatos de encabezado de mensajes de red y asignaciones de direcciones físicas.
- J2178-2 - Proporciona definiciones de parámetros de datos.
- J2178-3 - Define estándares para ID de tramas de mensajes de red para encabezados de byte único.
- J2178-4 - Define estándares para mensajes de red con tres encabezados de bytes.
- J2284-3 - Define capa física y enlace de datos CAN de 500 kbit/s.
- J2411 - Describe el protocolo GMLAN (single-wire CAN), usado en vehículos GM más recientes. A menudo accesible en el conector OBD como PIN 1 en vehículos GM más recientes.

2.3.4 Protocolo ISO para OBD-II

ISO, la Organización Internacional de Normalización, es una organización independiente, no gubernamental, fundada el 23 de febrero de 1947, que promueve estándares comerciales e industriales. Tiene su sede en Ginebra, Suiza, y desde el 2015 opera en 163 países.

Es el mayor creador mundial de estándares internacionales que facilitan el comercio mundial proporcionando estándares comunes entre las naciones. Se han establecido casi veinte mil normas que abarcan desde productos manufacturados y tecnología, hasta seguridad alimentaria, agricultura y atención sanitaria.

Normas ISO para OBD II:

- ISO 9141: Vehículos de carretera - Sistemas de diagnóstico. Organización Internacional de Normalización, 1989.
  - Parte 1: Requisitos para el intercambio de información digital.
  - Parte 2: Requisitos CARB para el intercambio de información digital.
  - Parte 3: Verificación de la comunicación entre el vehículo y la herramienta de barrido OBD II.
- ISO 11898: Vehículos de carretera - Red de área del controlador (CAN) Organización Internacional de Normalización, 2003.
  - Parte 1: Capa de enlace de datos y señalización física.
  - Parte 2: Unidad de acceso a medios de alta velocidad.
  - Parte 3: Interfaz de baja velocidad, tolerante a fallos y medio dependiente.
  - Parte 4: Comunicación en tiempo real.

- ISO 14230: Vehículos de carretera - Sistemas de diagnóstico - Protocolo de palabras clave 2000, Organización Internacional de Normalización, 1999.
  - Parte 1: Capa física.
  - Parte 2: capa de enlace de datos.
  - Parte 3: Capa de aplicación.
  - Parte 4: Requisitos para los sistemas relacionados con las emisiones.

- ISO 15031: Comunicación entre el vehículo y el equipo externo para diagnósticos relacionados con las emisiones, Organización Internacional de Normalización, 2010.
  - Parte 1: Información general y definición del caso de uso.
  - Parte 2: Orientación sobre términos, definiciones, abreviaturas y acrónimos.
  - Parte 3: Conector de diagnóstico y circuitos eléctricos relacionados, especificación y uso.
  - Parte 4: Equipo de prueba externo.
  - Parte 5: Servicios de diagnóstico relacionados con las emisiones.
  - Parte 6: Definiciones de códigos de problemas de diagnóstico.
  - Parte 7: Seguridad de enlace de datos.

- ISO 15765: Vehículos de carretera - Diagnóstico en redes de área del regulador (CAN). Organización Internacional de Normalización, 2004.
  - Parte 1: Información general.
  - Parte 2: Servicios de capa de red ISO 15765-2.
  - Parte 3: Implementación de servicios de diagnóstico unificado (UDS en CAN).
  - Parte 4: Requisitos para los sistemas relacionados con las emisiones.

2.4 Sistemas operativos para dispositivos móviles inteligentes

2.4.1 Android

Android es un sistema operativo móvil basado en Linux, elaborado por la compañía Google, y desarrollado para Smartphone, Tablet etc., con la idea de proporcionar a los fabricantes un sistema flexible y actualizado de bajo coste. Las aplicaciones para este sistema operativo están programadas en lenguaje Java.

Ha sido desarrollado con código abierto. El software abierto (en inglés *open-source software* u OSS) es aquel cuyo código fuente y otros derechos que normalmente son exclusivos para aquellos que poseen los derechos de autor, son publicados bajo una licencia de software compatible con la *Open Source Definition* o que forman parte del dominio público. Esto les permite a los usuarios utilizar, cambiar y mejorar el software, y redistribuirlo, ya sea en su forma modificada o en su forma no modificada.

En el año 2005, Google adquirió la empresa *Android Inc.* que diseñaba software para móviles. Posteriormente, en 2007 se creó un consorcio integrado por varias compañías tecnológicas como fabricantes de hardware, software, etc. para dispositivo móviles y que recibió el nombre de *An Open Handset Alliance Project,* que presentó la plataforma Android para móviles basada en la versión 2.6 de Kernel de Linux [5].

En cuanto a sus características, el sistema operativo se halla en una zona de memoria de sólo lectura por dos motivos: para evitar que el usuario lo dañe sin querer; y para que se sea fiel a las pequeñas modificaciones y aplicaciones integradas que los fabricantes suelen incluir en sus modelos.

Para actualizar un Smartphone Android se tienen varias opciones, siempre dependiendo de la operadora y del fabricante del dispositivo. Algunos permiten actualizar por medio de la conexión USB entre el Smartphone y el PC, y otros directamente descargando en el

dispositivo un archivo en la microSD y encendiendo el Smartphone. Sin embargo, es habitual que se actualice Android por OTA (*Over The Air*) o inalámbricamente.

Google Play es una tienda virtual de software que permite a los clientes descargar Apps, tanto gratuitas como de pago. También permite que los usuarios publiquen las aplicaciones que desarrollan.

La arquitectura de Android está distribuida en diferentes capas (Ilustración 2.6) que se describen a continuación:

- *Applications* (Aplicaciones): Las aplicaciones base incluyen un cliente de correo electrónico, programa de SMS, calendario, mapas, navegador, contactos y otros. Todas las aplicaciones están escritas en lenguaje de programación Java.

- *Application Framework* (Marco de trabajo de aplicaciones): Los desarrolladores tienen acceso completo a las mismos APIs del *framework* usados por las aplicaciones base. La arquitectura está diseñada para simplificar la reutilización de componentes; cualquier aplicación puede publicar sus capacidades y cualquier otra aplicación puede luego hacer uso de esas capacidades (sujeto a reglas de seguridad del *framework*). Este mecanismo permite que los componentes sean reemplazados por el usuario.

- *Libraries* (Bibliotecas): Android incluye un conjunto de bibliotecas de C/C++ usadas por varios componentes del sistema. Estas características se exponen a los desarrolladores a través del *framework* de aplicaciones de Android. Las bibliotecas, escritas en lenguaje C/C++, incluyen un administrador de pantalla táctil (*surface manager*), un *framework Open Core* (para el aprovechamiento de las capacidades multimedia), una base de datos relacional *SQLite*, una API gráfica *OpenGL ES 2.0 3D*, un motor de renderizado *WebKit*, un motor gráfico SGL, un protocolo de comunicación segura SSL y una biblioteca estándar de C, llamada "*Bionic*" y desarrollada por Google específicamente para Android a partir de bibliotecas estándar "*libc*" de BSD.

- *Android Runtime* (Funcionalidad en tiempo de ejecución): Android incluye un set de bibliotecas base que proporcionan la mayor parte de las funciones disponibles en las bibliotecas base del lenguaje Java. Cada aplicación Android corre su propio proceso, con su propia instancia de la máquina virtual *Dalvik*. *Dalvik* ha sido escrito de forma que un dispositivo puede correr múltiples máquinas virtuales de forma eficiente. *Dalvik* ejecuta

archivos en el formato *Dalvik Executable* (.dex), el cual está optimizado para memoria mínima.

- *Linux Kernel* (Núcleo Linux): Android dispone de un núcleo basado en *Linux* para los servicios base del sistema como seguridad, gestión de memoria, gestión de procesos, pila de red y modelo de controladores, y también actúa como una capa de abstracción entre el *hardware* y el resto de la pila de *software*.

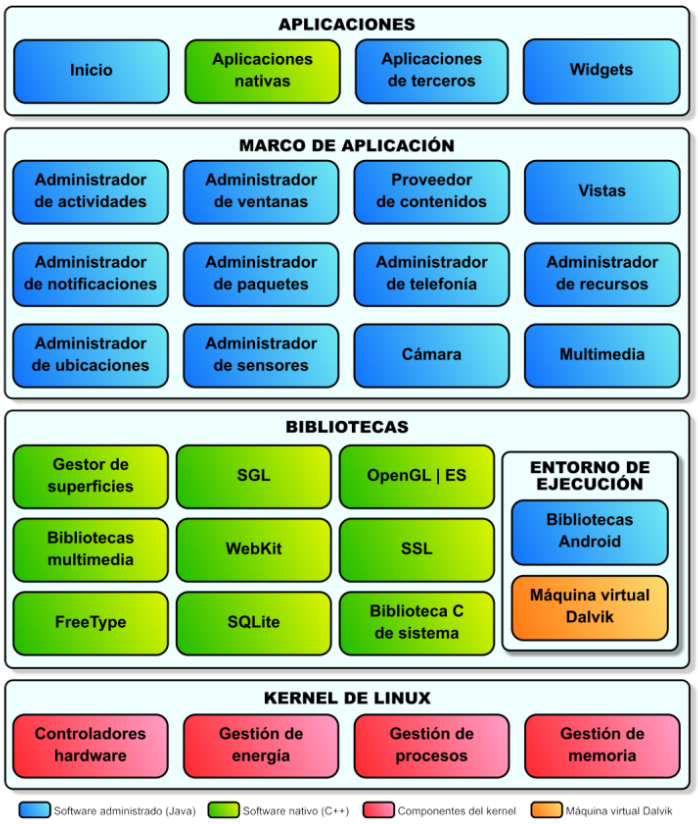


Ilustración 2.6. ***Arquitectura del sistema operativo Android***

Los desarrolladores informaron de la dificultad de mantener aplicaciones para versiones diferentes de Android, debido a problemas de compatibilidad entre la versión 1.5 y 1.6, especialmente, por diferencias de resolución entre los distintos teléfonos Android.

La evolución tan seguida de las versiones ha generado un problema llamado *desfragmentación,* que consiste en que dispositivos antiguos no soporten las nuevas versiones.

Estos problemas se hicieron patentes durante el concurso ADC2. Posteriormente, el rápido aumento de modelos de teléfonos basados en Android con diferentes capacidades de *hardware* complicaba el desarrollo de aplicaciones para todos los modelos de teléfonos Android.

Cada nueva versión de Android recibe el nombre de un postre en inglés (ver Ilustración 2.7)

Ilustración 2.7. ***Detalle de algunas versiones de Android***

Las aplicaciones se desarrollan en lenguaje Java, al que se le añade un paquete especial para dicho sistema operativo, denominado *Android Software Development Kit*. Este kit de desarrollo de software provisto por Google permite adaptar la programación a las características del sistema operativo Android [6].

Sin embargo, la situación parece haber mejorado "en parte", porque aún más del 54% de teléfonos con Android usan versiones 2.3.X (publicada en 2010). El principal problema de esto es a la hora de desarrollar aplicaciones nuevas con las APIs más recientes del SDK de Android, ya que se tiene que usar una versión de API compatible con 2.3, así las nuevas funcionalidades de APIs superiores, como NFC a partir de la versión Android ICS 4.0, no se pueden usar, al menos que se implementen a mano por no poder usar la API que hace uso de su funcionalidad.

2.4.2 iOS

El sistema operativo iOS (anteriormente denominado iPhone OS) ha sido desarrollado por Apple Inc. Concretamente, fue desarrollado para el Smartphone iPhone, siendo después aplicado a los dispositivos iPod Touch, iPad y el Apple TV.

La interfaz de usuario de iOS está basada en la manipulación directa, usando gestos multitáctiles (*multitouch*). Los elementos de control son deslizadores, interruptores y botones. Todos estos controles se caracterizan por una respuesta inmediata a las órdenes del usuario y con una interfaz fluida.

Proviene de Mac OS X, basado a su vez en Darwin BSD (plataforma de código abierto), que consiste en un sistema operativo tipo Unix.

La arquitectura del sistema operativo iOS (ver Ilustración 2.8) está basado en cuatros capas de abstracción: capa del núcleo del sistema operativo, capa de "Servicios Personales", capa de "Medios" y capa de "*Cocoa Touch*".

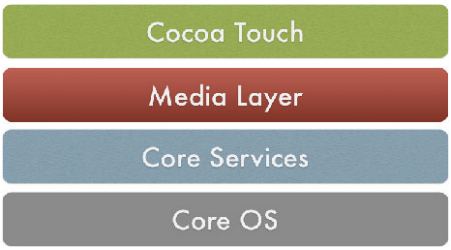


Ilustración 2.8. ***Arquitectura del sistema operativo iOS***

2.5 Sistemas de comunicación

En esta sección se van a introducir algunas alternativas para dotar a un sistema de procesamiento de capacidad de comunicación, de manera que se pueda intercambiar información con la unidad de procesamiento de forma inalámbrica.

2.5.1 Módulo Bluetooth HC-05

Se denomina Bluetooth al protocolo de comunicaciones diseñado especialmente para dispositivos de bajo consumo, que requieren corto alcance de emisión y basados en transceptores de bajo coste.

Bluetooth es una especificación industrial para Redes Inalámbricas de Área Personal (WPAN) que posibilita la transmisión de voz y de datos entre diferentes dispositivos, mediante un enlace por radiofrecuencia en la banda ISM de los 2,4 GHz.

Los principales objetivos que pretende conseguir son:

- Facilitar las comunicaciones entre equipos móviles y fijos.
- Eliminar los cables y conectores entre éstos.
- Ofrecer la posibilidad de crear pequeñas redes inalámbricas y facilitar la sincronización de datos entre equipos personales.

Los dispositivos que con mayor frecuencia utilizan esta tecnología pertenecen a sectores de las telecomunicaciones y la informática personal, como PDA, Smartphone, computadoras portátiles, ordenadores personales, impresoras o cámaras digitales.

El módulo de comunicación HC-05 (ver Ilustración 2.9) se puede emplear para conectar un sistema de procesamiento como Arduino por Bluetooth. Dicho protocolo está integrado de fábrica en la mayoría de dispositivos, portátiles, Tablets, y Smartphones, y su uso es independiente del sistema operativo [7].

Ilustración 2.9. ***Módulo Bluetooth***

2.5.2 BLE Shield

Bluetooth de baja energía, también denominada como Bluetooth LE, Bluetooth ULP (*Ultra Low Power*) y Bluetooth Smart, es una nueva tecnología digital de radio (inalámbrica) interoperable para pequeños dispositivos desarrollada por Bluetooth SIG.

En comparación con Bluetooth clásico, Bluetooth Smart está diseñado para proporcionar un consumo de energía y un coste considerablemente reducidos, manteniendo un rango de comunicación similar. Fue introducido originalmente bajo el nombre Wibree por Nokia en 2006. Se fusionó en el estándar principal de Bluetooth en 2010, con la adopción de la especificación básica Bluetooth versión 4.0. Es importante mencionar, que, a pesar de estar integrado con Bluetooth, los dispositivos de tipo Bluetooth tradicional y Bluetooth Low Energy no son compatibles y no pueden intercambiar información.

Los sistemas operativos móviles, como iOS, Android, Windows Phone y BlackBerry, así como macOS, Linux, Windows 8 y Windows 10, admiten de forma nativa Bluetooth Smart.

La Ilustración 2.10 muestra un ejemplo de una placa de desarrollo Bluetooth Low Energy, concretamente el BLE Shield desarrollado por Red Bear Lab. Esta placa se puede conectar directamente sobre una tarjeta Arduino y permite dotar a la misma de comunicación BLE.

El desarrollo de aplicaciones con este módulo resulta sencillo, pues proporciona repositorios tanto para Arduino como para iOS y para Android. Estos repositorios incluyen las bibliotecas necesarias para los sistemas mencionados, así como una batería de ejemplos que puede servir como punto de partida para construir cualquier aplicación deseada [8].

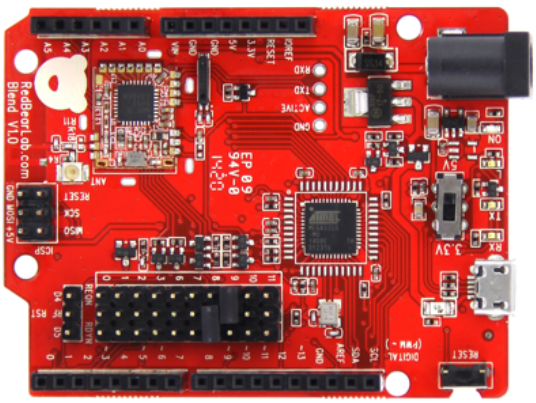


Ilustración 2.10. ***BLE Shield de Red Bear Lab***

2.5.3 Wifi

Fue desarrollado por la Wi-Fi Alliance y presentado en septiembre de 1997; Wi-Fi o WiFi es una tecnología para redes inalámbricas de área local con dispositivos basados en los estándares IEEE 802.11.

Los dispositivos compatibles con Wi-Fi pueden conectarse a Internet a través de una red WLAN y de un punto de acceso inalámbrico. El punto de acceso (o *hotspot*) tiene un alcance de unos 20 metros en interiores y un alcance mayor en entornos exteriores.

El ESP8266 (ver Ilustración 2.11) es un chip Wi-Fi de bajo coste con pila TCP/IP y capacidad de MCU (*Micro Controller Unit*) producida por el fabricante *Espressif Systems*. Con este módulo se puede desarrollar un dispositivo completo, ya que integra tanto el sistema de procesamiento como el módulo de comunicación WiFi [9].

Ilustración 2.11. ***Módulo WiFi ESP8266***

El chip fue distribuido con el módulo ESP-01, fabricado por AI-Thinker. Este pequeño módulo permite que los microcontroladores se conecten a una red Wi-Fi y realicen conexiones TCP/IP simples utilizando comandos de tipo Hayes.

Las aplicaciones para este módulo se pueden escribir con el IDE de Arduino utilizando su misma sintaxis.

El módulo descrito proporciona pocas líneas de E/S, pero existen otras placas de desarrollo con un mayor número de líneas.

Capítulo 3

Descripción del sistema

3.1 Introducción

En el capítulo anterior se han estudiado algunos de los dispositivos empotrados y plataformas disponibles para los diferentes módulos que componen el sistema.

De las alternativas propuestas, se ha seleccionado la plataforma Andorid, y como dispositivos empotrados a integrar en la plataforma: Arduino, módulo Bluetooth HC-05 y OBD-II Uart Hookup Guide.

Se ha decidido utilizar la plataforma Android por estar muy extendida y ser gratuita. Se trata de una plataforma innovadora con gran potencial y emprendimiento, que fomenta la creación de nuevos proyectos en varios continentes.

En relación a los dispositivos empotrados, la elección se debe a que Arduino es un sistema económico cuyo uso está muy extendido en el mundo académico y no académico, así como que su programación cuenta con grandes especificaciones y se distribuye con código abierto.

OBD to RS232 Interpreter es un sistema efectivo para la lectura de la unidad de mando de distintos modelos de vehículos, que permite la lectura de distintos parámetros del motor, la obtención de averías en tiempo real, y el borrado de las mismas.

El módulo Bluetooth HC-05 es compatible con Arduino y con teléfonos móviles de plataforma Android. Este módulo permite la transmisión inalámbrica de información de forma segura e instantánea entre los dispositivos involucrados en el sistema.

En el presente capítulo se ampliará la información ya expuesta en el anterior. Para ello, a continuación se describirá detalladamente los sistemas utilizados, profundizando en los aspectos y características determinantes en el desarrollo del este proyecto.

3.1.1 Arduino UNO

En este apartado se describe el *hardware* y el *software* de la placa Arduino UNO, así como el entorno de desarrollo necesario para la realización de un proyecto con Arduino.

La plataforma de desarrollo Arduino de computación física (*physical computing*) es de código abierto y está basada en una placa con un micro-controlador y un entorno de desarrollo para crear el código final que se puede ejecutar en la placa.

Se puede emplear Arduino para crear objetos interactivos, leyendo datos de una gran variedad de interruptores y sensores y controlar multitud de luces, motores y otros actuadores físicos. Los proyectos de Arduino pueden ser autónomos o comunicarse con un programa (*software*) ejecutable en un ordenador o un dispositivo móvil inteligente.

El lenguaje de programación de Arduino es una implementación de *Wiring*, otra plataforma de computación física, basada en el entorno de programación multimedia *Processing*.

Las principales ventajas de Arduino respecto a otros sistemas son las siguientes:

- Asequible: Las placas Arduino son más asequibles que otras plataformas de micro-controladores.

- Multi-Plataforma: El software de Arduino es compatible con los sistemas operativos Windows, Macintosh OSX y Linux.

- Entorno de programación simple y directo: El entorno de programación de Arduino es de fácil utilización para principiantes, a la par que suficientemente flexible para los usuarios avanzados.

- *Software* ampliable y de código abierto: El *software* Arduino está publicado bajo una licencia libre y preparado para ser ampliado por programadores experimentados. El lenguaje puede ampliarse a través de librerías escritas con el lenguaje C++.

- *Hardware* ampliable y de código abierto: Arduino está basado en los micro-controladores ATMEGA168, ATMEGA328 y ATMEGA1280. Los esquemáticos de los módulos son publicados con licencia *Creative Commons*, así que los diseñadores de circuitos pueden desarrollar su propia versión del módulo, ampliándolo u optimizándolo.

Ilustración 3.1. ***Vista de la placa Arduino UNO.***

La placa de Arduino UNO (ver Ilustración 3.1) está basada por el micro-controlador ATMEGA328. Proporciona 14 pines digitales de entrada/salida (de los cuales 6 se pueden utilizar como salidas PWM), 6 entradas analógicas, una conexión USB utilizada para cargar los programas en la placa y para la comunicación serie entre la placa y el ordenador (puede utilizarse como alimentación a la placa), un conector Jack DC de alimentación externa, un programador serie en circuito "In-circuit Serial Programmer" o "ICSP" y un botón de reset.

Resumen

En la siguiente tabla se resumen las principales características de la placa Arduino.

Característica	Descripción
Micro-controlador	ATMEGA328
Voltaje Entrada (Recomendado)	7 – 12 V
Pines E/S Digitales	14 (6 proporcionan salida PWM)
Pines Entrada Analógica	6
Corriente Pines E/S	40 mA
Corriente Pin 3,3 V	50 mA
Memoria Flash	32 KB (0,5 KB para el bootloader)
SRAM	2 KB
EEPROM	1 KB
Velocidad Reloj	16 MHz

Tabla 3.1. ***Resumen de las características de la placa Arduino.***

Memoria

El ATMEGA328 tiene 32 KB de Memoria Flash para almacenar código (de los cuales 0,5 KB se emplean en el bootloader). Tiene 2 KB de SRAM y 1 KB de EEPROM, que se puede leer y escribir con la librería EEPROM que proporciona la arquitectura Arduino.

Funciones Pines E/S

Cada uno de los 14 pines digitales (que trabajan con tensiones de 5 V) de Arduino UNO pueden ser utilizados como entrada o salida, mediante las funciones "digitalRead()", "digitalWrite()" y "pinMode()". Cada pin es apto para proporcionar o recibir un máximo de 40 mA, e integra una resistencia *pull-up* interna (desconectada por defecto) de 20 a 50 kΩ. Algunas funciones especiales de los pines son:

- Serie: Pines 0 (Rx) y 1 (Tx). Se utiliza para recibir (Rx) y de transmitir (TX) datos en serie usando niveles TTL (*Transistor-Transistor Logic*).

- Interrupciones Externas: Pines 2 y 3. Estos pines pueden ser configurados para activar una interrupción, que se pueden detectar tanto con flancos de subida como de bajada.

- PWM: Pines 3, 5, 6, 9, 10 y 11. Proporcionan salidas PWM (*Pulse Width Modulation*) de 8 bits.

- SPI: Pines 10 (SS), 11 (MOSI), 12 (MISO) y 13 (SCK) soportan la comunicación SPI empleando la librería SPI.

- Arduino UNO tiene 6 entradas analógicas (A0 – A5) de, 10 bits de resolución, por tanto, 1024 valores diferentes.

- TWI o I^2C: Pines A4 o SDA y pines A5 o SCL. Soportan comunicación TWI empleando la librería Wire.

- AREF: permite fijar una referencia positiva externa para los convertidores analógico-digitales. Por defecto se utiliza una referencia interna de 5 V.

- *Reset*: es un pin para resetear el micro-controlador, donde se agrega un botón de reinicio para los *Shields*.

Comunicación

La placa Arduino UNO soporta tanto comunicaciones con un ordenador, otro Arduino, u otros micro-controladores.

El ATMEGA328 provee comunicación serie UART TTL (5V), disponible en los pines digitales 0 (Rx) y 1 (Tx). Un FTDI FT232RL en la placa canaliza esta comunicación serie al USB y los drivers FTDI (incluidos con el software Arduino) proporcionan un puerto de comunicación virtual al software del ordenador. El monitor serie del software Arduino permite intercambiar datos de texto con su placa.

El ATMEGA328 también soporta comunicación I^2C (TWI) y SPI.

Programación

La placaArduino UNO puede ser programada con el software Arduino vía USB. El propio *software* dispone de los drivers necesarios para emular el puerto serie.

El ATMEGA328 de Arduino UNO dispone de un *bootloade*r pregrabado, que permite cargar nuevo código sin necesidad de un programador hardware externo. La comunicación se realiza mediante el protocolo original STK500.

Además, es posible programar el ATMEGA328 a través del conector ICSP (*In-Circuit Serial Programming*), sin necesidad de emplear el *bootloader,* pero en este caso se requiere de un programador externo o de otra placa Arduino configura como tal.

Protección de Sobrecargas del bus USB

La plataforma Arduino UNO posee un fusible que protege los puertos USB del ordenador de cortes y sobrecargas. Además, de la protección que ya incorporan la mayoría de los ordenadores, el fusible proporciona un nivel de protección adicional. Si se demandan más de 500 mA del puerto USB, el fusible automáticamente interrumpirá la conexión hasta que el corte o la sobrecarga sean eliminados.

Alimentación

La placa Arduino UNO puede ser alimentada a través de la conexión USB o con una fuente de alimentación externa, la selección es automática.

El rango recomendado para el correcto funcionamiento de la placa es entre 7 y 12 V. Aunque la placa puede funcionar con un suministro externo de 6 a 20 V. Si se proporcionan menos de 7 V, el pin de 5 V suministrará por debajo de su voltaje, de modo

que el micro-controlador no funcionará correctamente. Por otro lado, con una alimentación superior a 12 V, el regulador de voltaje se puede sobrecalentar y dañar la placa.

El terminal *Vin* es empleado para la alimentación de la placa. El voltaje suministrado comprende el rango de 7 a 12 V. Al emplear el conector de alimentación para alimentar la placa se puede utilizar el pin *Vin* como salida.

Al alimentar la placa a través del pin de salida de 5 V puede resultar dañada la placa, pues no está protegido frente a sobrecarga.

La salida de 3,3 V, está limitada a 50 mA de corriente máxima.

Los pines GND son de tierra.

El pin IOREF de la placa proporciona la referencia de tensión a los *Shields,* con la que opera el micro-controlador.

Mapeado entre ATMEGA168/328 y Arduino

A continuación se muestra el mapeado de pines entre el micro-controlador ATMEGA168/328 y la placa Arduino UNO (ver Ilustración 3.2).

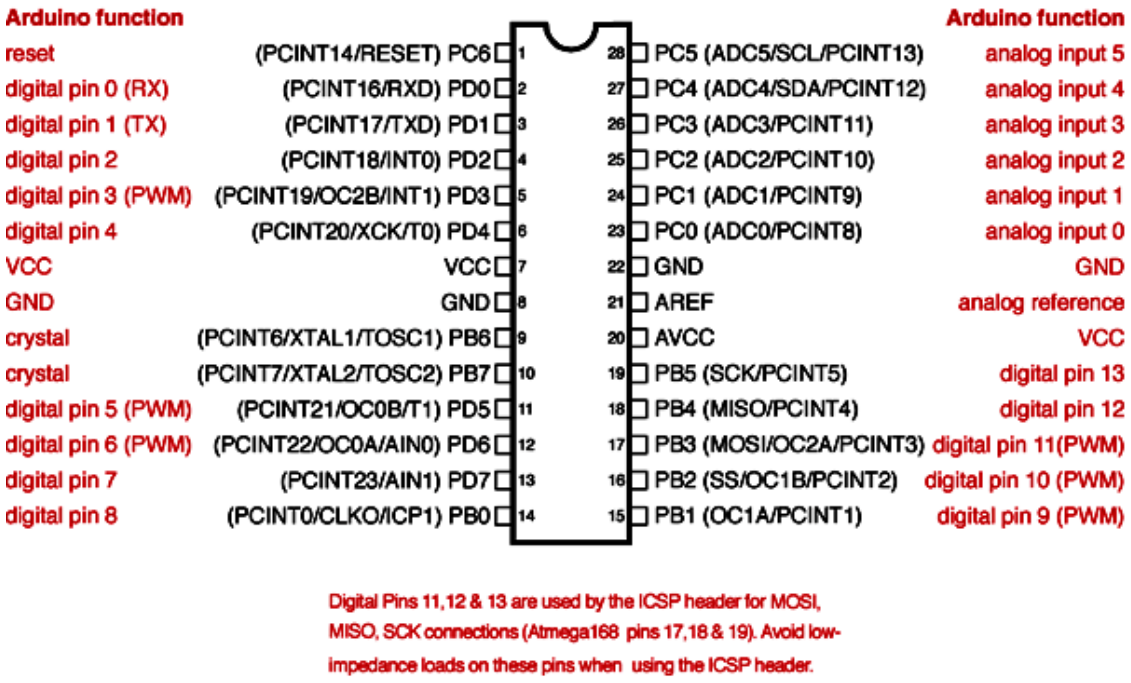


Ilustración 3.1. ***Mapeado de pines entre ATMEGA168/328 y Arduino.***

Librerías

Arduino ofrece una serie de librerías "estándar" basadas en C/C++ que se pueden importar al Sketck. Las principales bibliotecas *estándar* son [10]:

- EEPROM: Para leer y escribir en memorias "permanentes".
- Ethernet: Para conectar a Internet empleando el Ethernet Shield.
- Firmdata: Para comunicarse con aplicaciones en la computadora usando un protocolo estándar Serial.
- LiquidCrystal: Para controlar pantallas de cristal líquido (LCD)
- Servo: Para controlar servomotores.
- SoftwareSerial: Para la comunicación serial de cualquier pin digital.
- Stepper: Para controlar motores paso a paso.
- Wire: Interfaz de dos cables (TWI/I^2C), para enviar y recibir datos a través de una red de dispositivos y sensores.

Además de las anteriores librerías estándar, es posible disponer de otras en la red desarrolladas por la comunidad de usuarios.

Diseño Sketch

La estructura básica del lenguaje de programación Arduino se organiza en al menos dos partes o funciones que encierran bloques de declaraciones.

```
void setup() {
        Statements;
}
void loop(){
        Statements;
}
```

Ambas funciones son requeridas para que el programa funcione:

- setup(): La función "setup" debería contener la declaración de cualquier variable al comienzo del programa. Es la primera función a ejecutar en el programa.

Concretamente, es ejecutada una vez y usada por ejemplo para inicializar las comunicaciones serie.

• loop(): La función "loop" se ejecuta a continuación e incluye el código que se ejecuta continuamente leyendo entradas, activando salidas, etc. Esta función es el núcleo de todos los programas Arduino, y realiza la mayoría del procesamiento.

Entorno de Desarrollo

El entorno de desarrollo IDE (*Integrated Development Environment*) está escrito en Java y basado en Processing, avr-gcc y otros programas de código abierto.

El entorno de código abierto Arduino es un IDE multiplataforma, compatible con Windows, macOS y Linux.

También es un entorno gratuito y no requiere instalación (http://arduino.cc/en/Main/Software).

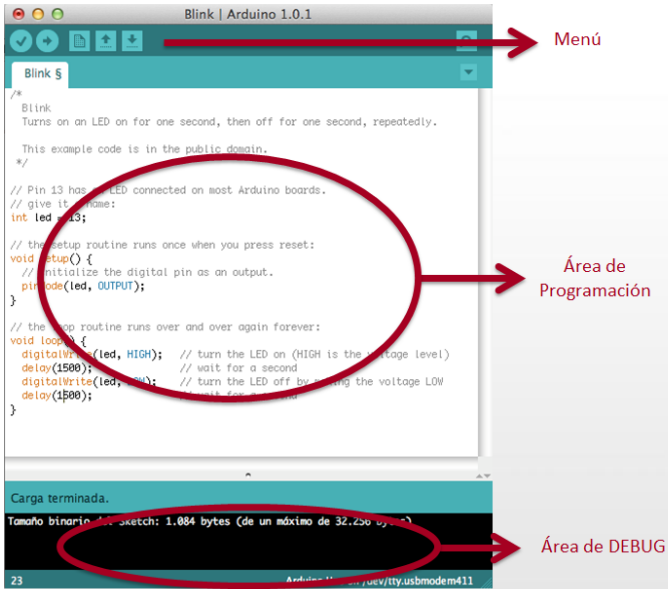


Ilustración 3.2. ***Vista entorno de Desarrollo Arduino.***

En el siguiente enlace se encuentra toda la información necesaria para la instalación del software en la plataforma Windows, Mac o Linux (http://arduino.cc/es/Guide/HomePage)

El software desarrollado con Arduino se conoce como sketches. Los sketches se escriben con un editor de texto y son guardados con la extensión ".ino".

Finalmente, la Ilustración 3.3 muestra los botones principales del entorno de Arduino.

Ilustración 3.3. ***Vista entorno de Desarrollo Arduino.***

3.2 Sistema de diagnóstico de vehículos

Las *Shields* (ver **¡Error! No se encuentra el origen de la referencia.**) son placas creadas para ser conectadas sobre la placa Arduino extendiendo sus capacidades, ya que los pines de sus puertos guardan una disposición de compatibilidad.

Ilustración 3.4. ***Shield OBD II Uart Hookup Guide con Arduino.***

Las diferentes Shields siguen la misma filosofía que el conjunto original: son fáciles de montar y de bajo coste económico.

Existe una gran variedad de *Shields* con diversa funcionalidad: control de motores, comunicaciones, prototipado rápido, etc.

3.2.1 Tarjeta OBD II Uart Hookup Guide

La placa UART OBD-II (ver **¡Error! No se encuentra el origen de la referencia.**) está conformada por los chips ELM327 y MCP2551, que permite al usuario acceder a los

protocolos CAN y OBD-II. El ELM327 es el chip controlador principal que se puede utilizar para convertir mensajes de cualquiera de los protocolos de OBD-II actualmente en uso a UART, y está basado en un núcleo de procesador de 16 bits. Se comunica con los transceptores CAN, ISO y J1850. El voltaje en la placa se regula a 5 V y 3,3 V para que todos los componentes funcionen correctamente. La placa se alimenta desde el conector DB9 [11].

La placa tiene dos puntos de conexión. El primero, en el borde exterior, es un conector de 6 pines compatible con una conversor FTDI de serie a USB. Sin embargo, sólo los pines Tx, Rx y GND están conectados en este conector, para permitir la comunicación UART. Hay un segundo conector de 8 pines cerca del conector DB9, que permite que el usuario acceda a la línea VBAT, el bus CAN, el bus LINE y el bus J1858, junto con el pin de tierra común.

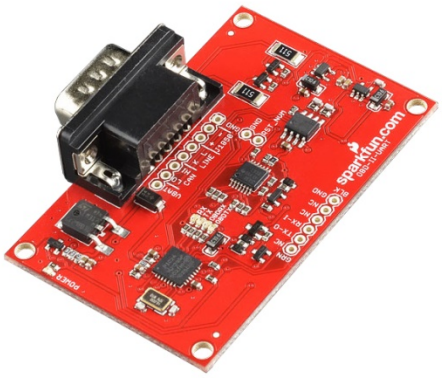


Ilustración 3.5. ***Tarjeta OBD II Uart Hookup Guide.***

El estándar OBD-II especifica el tipo de conector de diagnóstico y su pinout, los protocolos de señalización eléctrica disponibles y el formato de mensajería. También proporciona una lista de parámetros candidatos del vehículo a supervisar junto con la forma de codificar los datos de cada uno. Hay un pin en el conector que proporciona energía a la placa desde la batería del vehículo, lo que elimina la necesidad de conectarse a una fuente de alimentación auxiliar. Sin embargo, se recomienda que esté conectado a la fuente auxiliar para proteger los datos en el caso que un vehículo experimente una pérdida de energía eléctrica debido a un mal funcionamiento. Como resultado de esta estandarización, un solo dispositivo puede consultar el/los equipo/s de a bordo en cualquier vehículo.

Los códigos de diagnóstico de OBD-II (ver Ilustración 3.6) son de 4 dígitos, precedidos por una letra: P para motor y transmisión (tren de potencia), B para carrocería, C para chasis y U para red.

OBD-II proporciona acceso a los datos de la unidad de control del motor (ECU) y ofrece una valiosa fuente de información para solucionar problemas del vehículo. La norma SAE J1979 define un método para solicitar varios datos de diagnóstico y una lista de parámetros estándar que podrían estar disponibles desde la ECU. Los diversos parámetros que están disponibles se tratan mediante "números de identificación de parámetros" o PIDs que se definen en J1979.

Cada uno de los códigos de error EOBD consta de cinco caracteres: una letra, seguida de cuatro números. La letra se refiere al sistema que se está interrogando, Pxxxx se referiría al sistema del tren motriz. El siguiente carácter sería un 0 si cumple con el estándar EOBD. Así que debe ser como P0xxx. El siguiente carácter se referiría al sub-sistema. P00xx se refiere a medición de combustible y aire y controles de emisiones auxiliares. P01xx hace referencia a medición de combustible y aire. P02xx indica medición de combustible y aire (circuito inyector). P03xx hace referencia al sistema de encendido o fallo de encendido. P04xx indica controles de emisiones auxiliares. P05xx hace referencia a los controles de velocidad del vehículo y sistema de control de ralentí. Los códigos P06xx indican errores en el circuito de salida de la computadora. Otros códigos son el P07xx y el P08xx, que hacen referencia a la transmisión. Los dos caracteres siguientes se refieren al fallo individual dentro de cada subsistema.

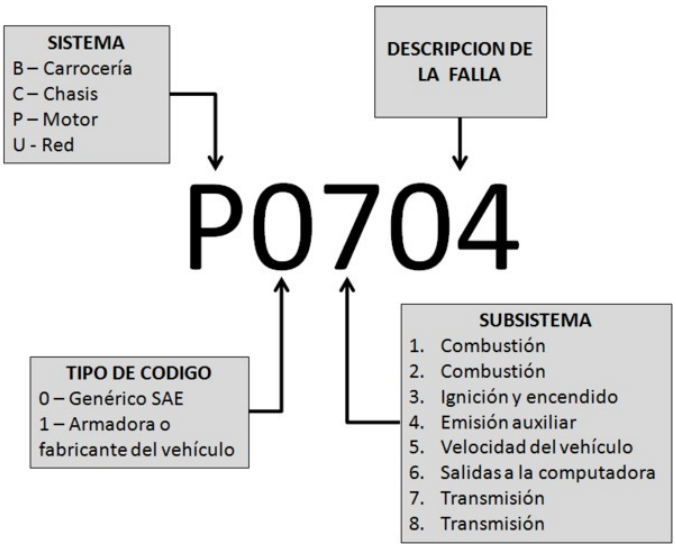


Ilustración 3.6.*Cuadro de fallos ELM327.*

ELM327

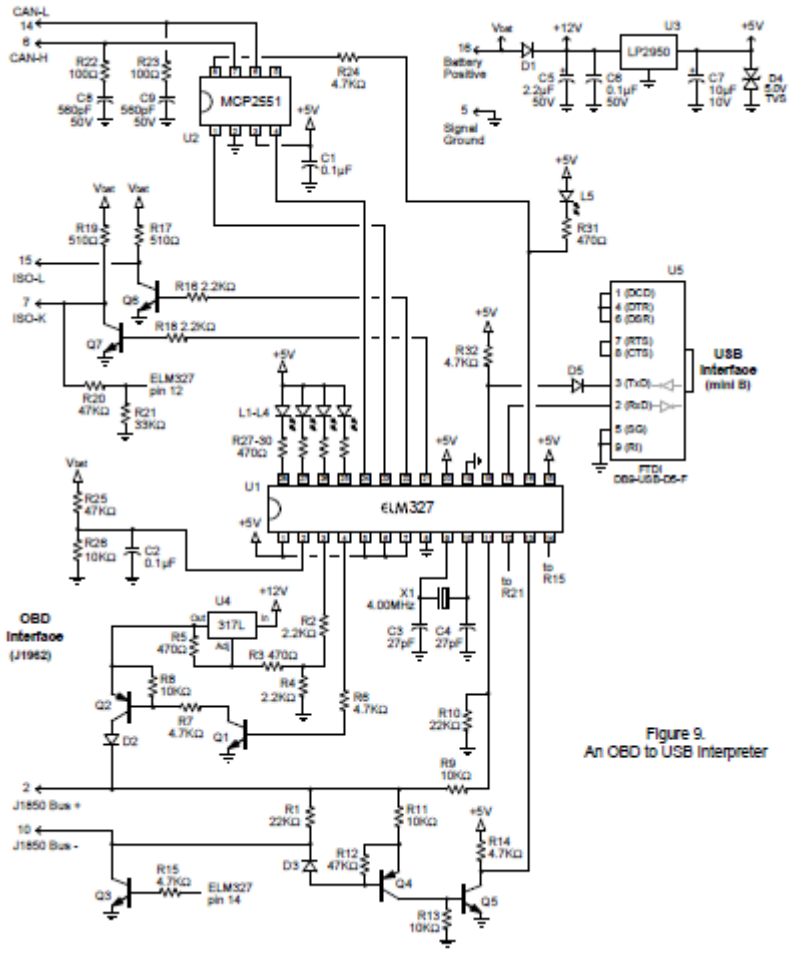


Ilustración 3.7. ***Esquema eléctrico OBD to USB Interpreter.***

El esquema anterior (ver **¡Error! No se encuentra el origen de la referencia.**) muestra los componentes electrónicos que integra la placa *OBD II Uart Hookup Guide* cuyo principal dispositivo es el microchip *ELM327*.

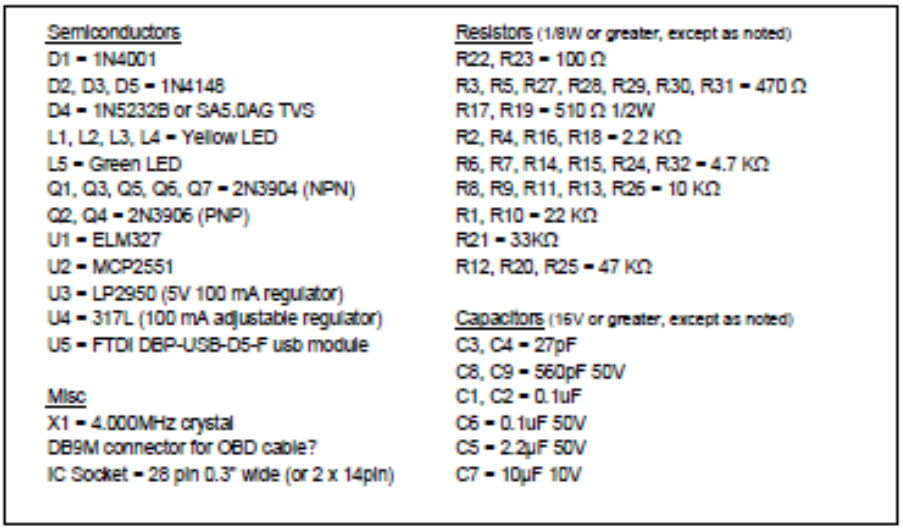
El regulador LP2950 limita la corriente disponible a unos 100 mA, y funciona como una caja de seguridad contra sobre intensidades.

Los cuatro LEDs mostrados (en las patillas 25 a 28) muestran cuando está activo el circuito.

Los pines RS232 (17 y 18) están conectados al módulo FTDI USB. Este módulo hace que sea muy fácil conectar el circuito ELM327 a un ordenador.

El diodo D5 y la resistencia R32 han sido añadidas a la interfaz para evitar las retroalimentaciones desde el USB en el ELM327.

Finalmente, el circuito de oscilación está formado por el cristal (conectado entre los pines 9 y 10) de 4 MHz y los condensadores estándar de 27pF.



Semiconductors
D1 = 1N4001
D2, D3, D5 = 1N4148
D4 = 1N5232B or SA5.0AG TVS
L1, L2, L3, L4 = Yellow LED
L5 = Green LED
Q1, Q3, Q5, Q6, Q7 = 2N3904 (NPN)
Q2, Q4 = 2N3906 (PNP)
U1 = ELM327
U2 = MCP2551
U3 = LP2950 (5V 100 mA regulator)
U4 = 317L (100 mA adjustable regulator)
U5 = FTDI DBP-USB-D5-F usb module

Misc
X1 = 4.000MHz crystal
DB9M connector for OBD cable?
IC Socket = 28 pin 0.3" wide (or 2 x 14pin)

Resistors (1/8W or greater, except as noted)
R22, R23 = 100 Ω
R3, R5, R27, R28, R29, R30, R31 = 470 Ω
R17, R19 = 510 Ω 1/2W
R2, R4, R16, R18 = 2.2 KΩ
R6, R7, R14, R15, R24, R32 = 4.7 KΩ
R8, R9, R11, R13, R26 = 10 KΩ
R1, R10 = 22 KΩ
R21 = 33KΩ
R12, R20, R25 = 47 KΩ

Capacitors (16V or greater, except as noted)
C3, C4 = 27pF
C8, C9 = 560pF 50V
C1, C2 = 0.1uF
C6 = 0.1uF 50V
C5 = 2.2µF 50V
C7 = 10µF 10V

Ilustración 3.8.*Componentes electrónicos de la placa OBD.*

El esquema eléctrico del ELM327 también muestra como el chip es alimentado del vehículo a través de los pines OBD 16 y 5. Además, dispone de un diodo de protección y de filtrado capacitivo en el regulador de tensión de cinco voltios. Cuando el vehículo no dispone del pin 5, se utilizará el pin 4.

Finalmente, para terminar esta sección presentar la definición SAE J1962 del pinout del conector OBD (ver Ilustración 3.9 y Tabla 3.2Tabla 3.2)

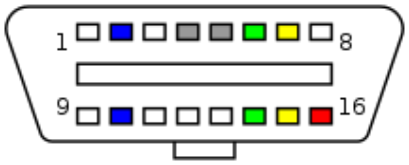


Ilustración 3.9*. Pinout OBD-II diagnostic connector.*

1	Manufacturer discretion: • GM: J2411 GMLAN/SWC/Single-Wire CAN • VW/Audi/BMW: Switched +12V to tell a scan tool whether the ignition is on. • Ford, FIAT: Infotainment CAN High	**9**	Manufacturer discretion: • BMW: TD (Tachometer Display) signal aka engine RPM signal. • GM: 8192 bit/s ALDL where fitted. • Ford: Infotainment CAN-Low
2	Bus Positive Line of SAE J1850 PWM and VPW	**10**	Bus Negative Line of SAE J1850 PWM only (not SAE J1850 VPW)
3	Manufacturer discretion: • Ford: DCL(+) Argentina, Brazil (pre OBD-	**11**	Manufacturer Discretion: • Ford: DCL(-) Argentina, Brazil (pre

	II) 1997-2000, USA, Europe, etc. • Ford: Medium Speed CAN-High[I] • Chrysler: CCD Bus(+)		OBD-II) 1997-2000, USA, Europe, etc. • Ford: Medium Speed CAN-Low[I] Chrysler: CCD Bus(-)
4	Chassis ground	**12**	Manufacturer discretion:
5	Signal ground	**13**	Manufacturer discretion: • Ford: FEPS – Programming PCM voltaje
6	CAN-High (ISO 15765-4 and SAE J2284)	**14**	CAN-Low (ISO 15765-4 and SAE J2284)
7	K-Line of ISO 9141-2 and ISO 14230-4	**15**	L-Line of ISO 9141-2 and ISO 14230-4
8	Manufacturer discretion: • BMW: Second K-Line for non OBD-II (Body/Chassis/Infotainment) systems. • FIAT: Infotainment CAN-Low.	**16**	Battery voltage: • Type "A" 12V/4A • Type "B" 24V/2ª

Tabla 3.2. ***Descripción de los pines del conector OBD-II***

Resumen

En la siguiente tabla (Tabla 3.3) se resumen las principales características de la tarjeta OBD II Uart Hookup Guide, comentadas con anterioridad.

Característica	Descripción
Micro-controlador	ELM327
Voltaje Entrada (Recomendado)	12 V
Velocidad interfaz UART	38 bps a 10 Mbps

Tabla 3.3. ***Resumen de las características de la placa OBD II Uart Hooup Guide.***

3.3 Sistemas operativos para dispositivos móviles inteligentes

3.3.1 Android

Instalación del Entorno de Desarrollo

Para ello, hay que descargar el paquete ADT en la página oficial de Android *Developer* (http://developer.android.com/sdk/index.html). Dicho paquete incluye todo lo necesario para comenzar a desarrollar aplicaciones para Android.

Ejecutar el archivo "*eclipse.exe*" para lanzar la aplicación. La primera vez que arranque, solicitará que se le indique un directorio de trabajo donde se guardarán los proyectos (marcando en "*Use this as the default and do not ask again*" no preguntará de nuevo por el directorio de trabajo).

El desarrollo de programas para Android emplea generalmente el lenguaje de programación Java y el conjunto de herramientas de desarrollo (SDK, *Software Development Kit*) [12, 13].

El SDK de Android incluye un conjunto de herramientas de desarrollo. Comprende un depurador de código, biblioteca, un simulador de teléfono basado en QEMU, documentación, ejemplos de código y tutoriales.

La plataforma integral de desarrollo (IDE, *Integrated Development Environment*) soportada oficialmente es Eclipse junto con el complemento ADT (*Android Development Tools Plugin*). Eclipse es un programa informático compuesto por un conjunto de herramientas de programación de código abierto multiplataforma para desarrollar, lo que se conoce como "Aplicaciones de Cliente Enriquecido".

Las actualizaciones del SDK están coordinadas con el desarrollo general de Android. El SDK soporta también versiones antiguas de Android, por si los programadores necesitan instalar aplicaciones en dispositivos ya obsoletos o más antiguos. Las herramientas de desarrollo son componentes descargables, de modo que una vez instalada la última versión, pueden instalarse versiones anteriores y hacer pruebas de compatibilidad [5].

Una aplicación Android está compuesta por un conjunto de ficheros empaquetados en formato ".apk".

Componentes de Android

El fichero Manifest, perteneciente al archivo AndroidManifest.xml, proporciona la información esencial que el sistema de la aplicación Android debe conocer para ejecutar el código de la aplicación. Algunas especificaciones del Manifest son:

- Nombre del paquete Java que será el identificador unívoco de la aplicación.
- Componentes de la aplicación y su capacidad, de modo que el sistema Android conozca los componentes y sus condiciones para su lanzamiento.
- Procesos que hospedarán los componentes de la aplicación.
- Permisos.

- Nivel mínimo de API para la ejecución de la aplicación.
- Bibliotecas necesarias que deben "linkarse" para que la aplicación funcione.

En la arquitectura de Android, las aplicaciones se desarrollan en código Java, mientras que bibliotecas nativas, linux Kernel, y el hardware se desarrollan en código C [14, 15].

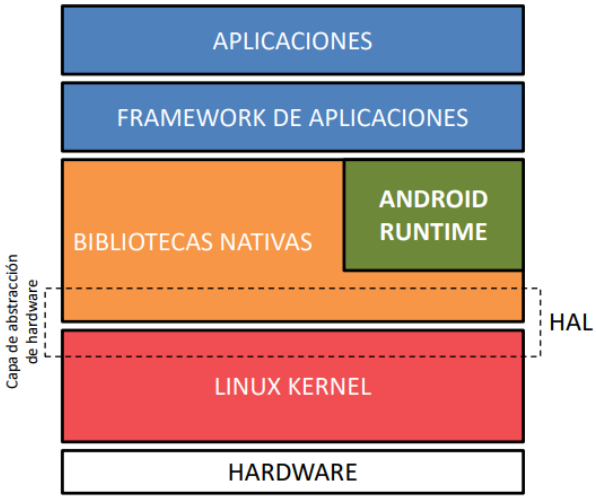


Ilustración 3.10. ***Arquitectura de Android***

Interfaces de usuario en Android

Para crear un diseño centrado en el usuario, la pantalla resultante de la aplicación está constituida por dos planos: foregroud (primer plano) y el background. Las aplicaciones con componentes foregroud tienen lugar prioritario en la pantalla, para ser más accesibles al usuario que las está utilizando. Las pantallas que presentan información y controles presentan una jerarquía de vistas.

Cada pantalla muestra una parte de la interfaz, cuyos elementos pueden crearse en tiempo de diseño o ejecución, o bien pueden modificarse en tiempo de ejecución.

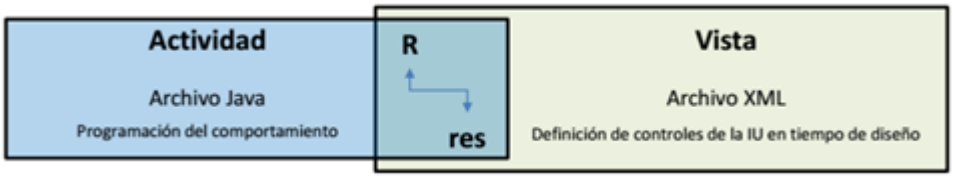


Ilustración 3.11. Componentes de la interfaz de usuario

La clase accesible desde código Java permite acceder a los recursos de la carpeta "res" que contiene los recursos de la aplicación.

Las actividades, componentes que interactúan con el usuario, permiten crear una ventana en la que se puede desplegar una interfaz de usuario, compuesta por una jerarquía de vistas. Para crear la actividad es necesario implementar una clase Java que extienda (derive o herede) de la clase Activity. Esta nueva clase puede incluir métodos nuevos, heredar los métodos existentes (y usarlos), y redefinir los métodos necesarios.

La actividad tiene un ciclo de vida consistente en pasar por diferentes estados, el paso de un estado a otro produce un evento que es manejado mediante la redefinición del método.

Layouts

La carpeta layouts almacena archivos xml que contienen descripciones de los controles de una jerarquía de vistas. Un layout define la estructura visual de una interfaz de usuario que mostrará una actividad o un "app widget".

Existen dos posibilidades para declarar un layout:

- Declarando los controles (elementos de la interfaz de usuario) en un archivo xml.
- Instanciando los elementos del layout en tiempo de ejecución. Permite modificar los controles declarados en un xml, o bien crearlos programáticamente, usando objetos View y ViewGroup.

Al compilar la aplicación cada archivo xml se compila sobre un recurso de tipo View, que pude ser cargado desde el código de una actividad usando el método **setContentView**, pasando como parámetro la referencia correspondiente en **R.layout**.

Los controladores de la interfaz son de dos tipos:

- Contenedores, elementos visuales que permiten almacenar y encajar otros elementos en la interfaz **ViewGroup**.
- Controles propiamente dichos, elementos visuales que permiten la interacción del usuario con la aplicación **View**.

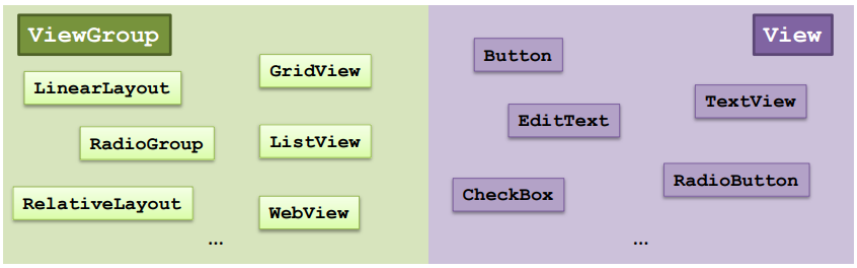


Ilustración 3.12. ***Controladores de la interfaz***

3.4 Sistemas de comunicación

3.4.1 Introducción

Como se mencionó en el capítulo anterior, el sistema de comunicación empleado será el módulo Bluetooth HC-05, que viene configurado de fábrica como esclavo.

3.4.2 Módulo HC-05 Bluetooth compatible con Arduino

Está integrado por dos componentes. El módulo de Bluetooth en sí, y la base a la que va soldado y que contiene el resto de circuitería.

Los dispositivos que incorporan este protocolo pueden comunicarse cuando se encuentran dentro de su alcance. Las comunicaciones se realizan por radiofrecuencia, de forma que los dispositivos no tienen que estar alineados y pueden incluso estar distantes en función de la potencia de transmisión. Los dispositivos se clasifican como "Clase 1", "Clase 2" o "Clase 3" (ver Tabla 3.4) en referencia a su potencia de transmisión, siendo totalmente compatibles los dispositivos de una clase con los de las otras [16].

Clase	Potencia máxima permitida (mW)	Potencia máxima permitida (dBm)	Alcance (aproximado)
Clase 1	100 mW	20 dBm	~100 metros
Clase 2	2.5 mW	4 dBm	~10 metros
Clase 3	1 mW	0 dBm	~1 metro

Tabla 3.4. ***Potencia Transmisión Dispositivos Bluetooth***

La cobertura efectiva de un dispositivo de clase 2 se extiende cuando se conecta a un transceptor de clase 1. Debido a la mayor sensibilidad y potencia de transmisión del dispositivo de clase 1, es decir, la mayor potencia de transmisión del dispositivo de clase 1 permite que la señal llegue con energía suficiente hasta el de clase 2. Por otra parte la mayor sensibilidad del dispositivo de clase 1 permite recibir la señal del otro pese a ser más débil.

Versión	Ancho de banda
Versión 1.2	1 Mbit/s
Versión 2.0 + EDR	3 Mbit/s
Versión 3.0 + HS	24 Mbit/s
Versión 4.0	24 Mbit/s

Tabla 3.5. ***Clasificación Dispositivos Bluetooth según su Ancho de Banda***

La especificación de Bluetooth define un canal de comunicación a un máximo 720 Kbit/s (1 Mbit/s de capacidad bruta) con rango óptimo de 10 m (opcionalmente 100 m con repetidores) (ver Tabla 3.5.)

Opera en la frecuencia de radio de 2,4 a 2,48 GHz con amplio espectro y saltos de frecuencia con posibilidad de transmitir en *full dúplex,* con un máximo de 1600 saltos por segundo. Los saltos de frecuencia se dan entre un total de 79 frecuencias con intervalos de 1 MHz; esto permite dar seguridad y robustez.

La potencia de salida para transmitir a una distancia máximo de 10 metros es de 0 dBm (1 mW), mientras que la versión de largo alcance transmite entre 20 y 30 dBm (entre 100 mW y 1 W)

Para lograr alcanzar el objetivo de bajo consumo y bajo costo se ideó una solución que se puede implementar en un solo chip utilizando circuitos CMOS.

El *hardware* que compone el dispositivo *Bluetooth* está compuesto por dos partes:

- Un dispositivo de radio, encargado de modular y transmitir la señal
- Un controlador digital, compuesto por una CPU, un procesador de señales digitales llamado *Link Controller* y de las interfaces con el dispositivo anfitrión

El *Link Controller* se encarga del procesamiento de la banda base y del manejo de los protocolos ARQ y FEC de la capa física; además, se encarga de las funciones de transferencia tanto asíncrona como síncrona, la codificación de audio y el cifrado de datos.

La CPU del dispositivo se encarga de las instrucciones relacionadas con *Bluetooth* en el dispositivo anfitrión, para así simplificar su operación. Para ello, sobre la CPU corre un *software* denominado *Link Manager* cuya función es la de comunicarse con otros dispositivos por medio del protocolo LMP.

Entre las tareas realizadas por el *Link Controller* y el *Link Manager*, destacan las siguientes:

- Envío y Recepción de Datos
- Paginación y Peticiones
- Establecimiento de conexiones
- Autentificación

- Negociación y establecimiento de tipos de enlace
- Establecimiento del tipo de cuerpo de cada paquete
- Establecer el dispositivo en modo *sniff* o *hold*

El módulo Bluetooth HC-05, es un Blueetooth V2, viene configurado de fábrica como esclavo, aunque se puede configurar como maestro, también permite cambiarle el nombre, código de vinculación velocidad, etc.

El módulo Bluetooth HC-05 funciona como esclavo cuando espera que un dispositivo Bluetooth maestro se le conecte, generalmente se utiliza cuando se necesita comunicarse con un dispositivo PC o teléfono (que se comportan como dispositivos maestros).

El módulo Bluetooth HC-05 se comporta como maestro cuando es el dispositivo que inicia la conexión. Un dispositivo maestro sólo se puede conectarse con un dispositivo esclavo. Generalmente se utiliza este modo para comunicarse entre módulos Bluetooth. Pero es necesario antes especificar con que dispositivo se tiene que comunicar.

El módulo HC-05 viene por defecto configurado del siguiente modo:

- Modo o role: *Esclavo*
- Nombre por defeco: *HC-05*
- Código de emparejamiento por defecto: *1234*
- La velocidad por defecto (baud rate): *9600*

Son cuatro los estados del módulo HC-05, se detallan a continuación con sus características:

- Estado Desconectado:
  - Comienza cuando se inicia la alimentación del módulo sin haberse establecido una conexión Bluetooth con ningún otro dispositivo
  - El LED del módulo en este estado parpadea rápidamente
  - En este estado el módulo no puede interpretar los comandos AT
- Estado Conectado o de comunicación
  - Entra a este estado cuando se establece una conexión con otro dispositivo Bluetooth
  - El LED hace un doble parpadeo
  - Todos los datos que se ingresen al HC-05 por el Pin RX se trasmiten por Bluetooth al dispositivo conectado, y los datos recibidos se devuelven por el pin TX
- Modo AT 1
  - Para entrar a este estado después de conectar y alimentar el módulo es necesario presionar el botón del HC-05
  - El LED del módulo en este estado parpadea rápidamente igual que en el estado desconectado

- Modo AT 2
  - Para entrar a este estado es necesario mantener presionado el botón en el instante de alimentar el módulo, es decir el módulo debe encenderse con el botón presionado, después de haber encendido se puede soltar y permanecerá en este estado.
  - En este estado, el envío de comandos AT tiene que hacerse a la velocidad de 38400 baudios
  - El LED del módulo en este estado parpadea lentamente

El envío de comandos AT se pueden realizar de modo directo entre los dispositivos maestro-esclavo, o de modo indirecto empleando una placa Arduino entre ambos.

Ilustración 3.13. ***Modo indirecto de envío de comandos AT.***

El módulo Bluetooth requiere un puerto serie de la placa Arduino. Por tanto, el módulo empleado de Bluetooth no permite el uso del puerto serie en las placas modelo Uno, Mini, y Nano. En el modelo Mega no existe este problema, porque incorpora cuatro puertos de serie.

Si realmente son necesarias ambas comunicaciones se puede recurrir al empleo de la librería *SoftSerial* para establecer una comunicación de puerto de serie por cualquier pareja de pines digitales, aunque supondrá un coste adicional de tiempo de proceso en Arduino.

La conexión (ver Ilustración 3.) es sencilla, alimentado mediante los pines *Vcc* y *GND*. Posteriormente se conectan el *TXD* (pin de transmisión) y *RXD* (pin de recepción) a los opuestos de la placa Arduino (cada *TXD* a un *RXD*) [17].

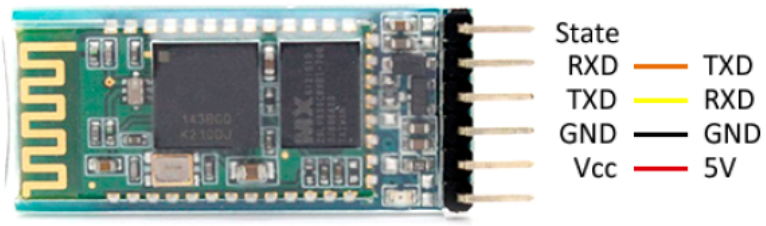


Ilustración 3.15. ***Módulo HC-05 Bluetooth.***

El esquema visto desde Arduino sería (ver Ilustración 3.) el siguiente:

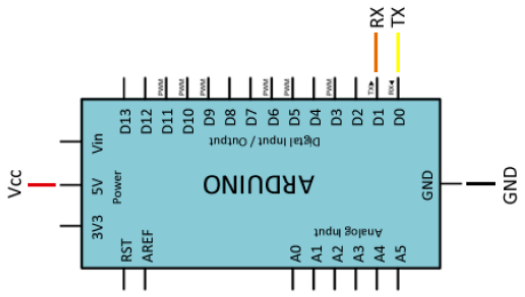


Ilustración 3.16. ***Esquema del módulo HC-05 Bluetooth.***

Los pines mostrados en el esquema son:

- *Vcc*, Voltaje positivo de alimentación con un rango de funcionamiento entre 3,3 V y 6 V, aunque existan módulos que sólo soportan voltajes de 3,3 V.
- *GND*, Voltaje negativo de alimentación conexión al GND del Arduino o de la placa utilizada.
- *Tx*, Pin de Transmisión de datos del módulo HC-05 con el PC o Móvil mediante Bluetooth. Este pin debe ir conectado al pin RX del Arduino.
- *Rx*, pin de Recepción: a través de este pin el módulo HC-05 recibirá los datos del Arduino los cuales se transmitirán por Bluetooth, este pin va conectado al Pin TX del Arduino.

Resumen

En la siguiente tabla (ver Tabla 3.6) se resumen las principales características del módulo HC-05 comentadas con anterioridad.

Característica	Descripción
Modo o role	Esclavo
Nombre por defeco	*HC-05*
Código de emparejamiento por defecto	*1234*
La velocidad por defecto Modo AT 1 (baud rate)	*9600*
La velocidad en Modo AT 2	*38400*
Ancho de banda de Versión 2.0+ EDR	1 Mbit/s

Tabla 3.6. ***Resumen de las características del módulo Bluetooth HC- 05.***

Cargar programa en Arduino con módulo Bluetooth HC-05

Si se usa la UART que proporciona la placa Arduino UNO (Tx y Rx), hay que tener la precaución de desconectar el pin Rx con el módulo Bluetooth antes de cargar un nuevo sketch en la placa de Arduino UNO.

Capítulo 4

Caso de estudio

4.1 Introducción

En el capítulo anterior se describió tanto el *hardware* como el *software* necesario para la realización de este Proyecto Fin de Carrera.

En este capítulo se detallará la implementación del *hardware* y el *software* desarrollado para culminar la Arquitectura y Diseño del sistema de diagnóstico de vehículos.

4.2 Descripción general de la arquitectura

La arquitectura del caso de estudio (ver Ilustración 4.1) está formado por el *Gadget* compuesto por los módulos empotrados de bajo coste: placa OBD II Uart Hooup Guide con microprocesador ELM327, placa Arduino, módulo Bluetooth HC-05, adaptador protoshield, un Smartphone con sistema operativo Android, y un vehículo con conector OBD. La conexión y alimentación eléctrica entre el *Gadget* y la unidad de mando (ECU) del vehículo se realiza con un cable OBD, mientras que la conexión entre el *Gadget* y el Smartphone es inalámbrica empleando Bluetooth.

Ilustración 4.1. ***Arquitectura del caso de estudio***

4.3 Descripción del hardware desarrollado

En este apartado (ver Ilustración 4.2) se describirán más en detalle las conexiones realizadas entre los distintos elementos *hardware* que componen el *Gadget*. La alimentación del *Gadget* se realiza por el OBD II Uart Hooup Guide ELM327, que recibe la energía del automóvil analizado a través de los pines 5 y 16 del conector mencionado. La placa dispone de un diodo de protección y de filtrado capacitivo en el regulador de tensión de 5 V. Cuando el vehículo no disponga del pin 5, se utilizará el pin 4. La placa Arduino UNO se alimenta a través de su pin *Vin*, cuya conexión entre la placa OBD II Uart Hooup Guide ELM327 y la placa Arduino se realiza mediante una placa protoshield. A su vez, desde el pin 5 V de la placa Arduino se alimenta el módulo Bluetooth HC-05 utilizando la misma placa protoshield.

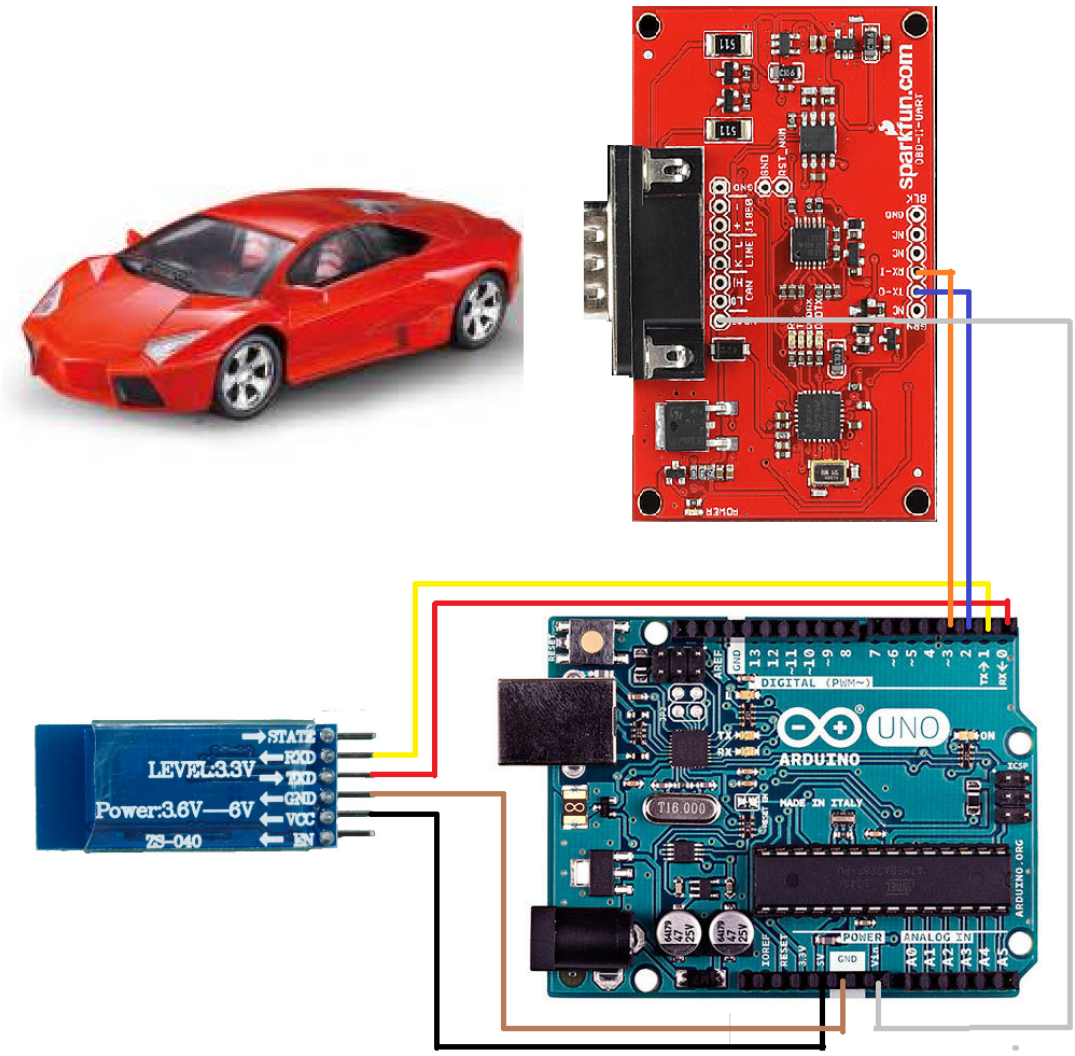


Ilustración 4.2. ***Conexiones de los elementos del hardware***

La conexión es inalámbrica entre el dispositivo maestro (PC, Smartphone, etc.) y el módulo Bluetooth HC-05. La particularidad de este tipo de conexión es que en ambos dispositivos se debe mantener el mismo tipo de configuración en cuanto a bits de inicio, bits de parada, velocidad y paridad, ya que dicha conexión inalámbrica aparece como un puerto serie emulado.

La UART toma los bytes de los datos y los transmite de forma secuencial al destino, donde una segunda UART reensambla los bits en bytes completos.

Para este tipo de conexión inalámbrica, la configuración de la UART utilizada es de 1 bit de inicio, 1 bit de parada, 9600 bps de velocidad y sin paridad.

A continuación, se procede a la descripción del protoshield. El protoshield sirve de adaptador entre la placa OBD II Uart Hooup Guide ELM327, la placa Arduino y el módulo Bluetooth HC-05. Para la conexión entre la placa OBD II Uart Hooup Guide

ELM327 y la placa Arduino, ha sido necesario unir externamente los pines en el protoshield (ver Tabla 4.1), configurándose una UART en los pines 2 y 3, mientras que el pin *VBAT* de alimentación de la placa OBD II Uart Hooup Guide ELM327 se conecta con el pin *Vin* de potencia de la placa Arduino.

Protoshield-ELM327 OBD	Protoshield-Arduino
Pin Tx-0	Pin 2
Pin Rx-1	Pin 3
Pin VBAT	Pin Vin

Tabla 4.1. ***Conexión pines placa protoshield con ELM327 OBD***

La conexión de la placa Arduino con el módulo Bluetooth HC-05 se consigue uniendo en el protoshiel los siguientes pines (ver Tabla 4.2). Se conectan cruzados los pines de emisión (pin *Tx*) y recepción (pin *Rx*) de datos de la placa Arduino y el módulo Bluetooth. El pin de alimentación de tensión *VCC* del módulo Bluetooth HC-05 va conectado al pin de alimentación de 5V de la placa Arduino. Los pines de masa o tierra van conectados entre sí en ambas placas.

Protoshield-Bluetooth HC-05	Protoshield-Arduino
Pin TxD	Pin Rx-0
Pin RxD	Pin Tx-1
Pin VCC	Pin 5V
Pin GND	Pin GND

Tabla 4.2. ***Conexiones pines placa protoshield con Bluetooth***

Finalmente, la placa protoshield queda como se muestra a continuación (ver Ilustración 4.3).

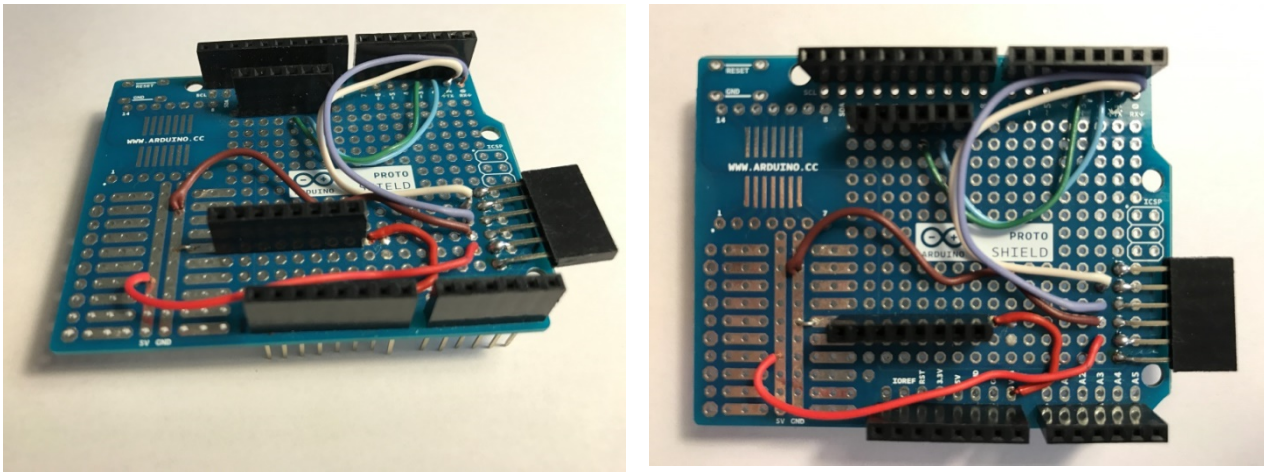


Ilustración 4.3. ***Placa Protoshield***

4.4 Descripción del software desarrollado

En este apartado se describirán el Sketch Arduino y la aplicación Android realizada, así como su código fuente.

4.4.1 Sketch Arduino

Como se comentó en el capítulo 3, la estructura básica de un Sketch de Arduino se basa en al menos 2 funciones setup() y loop() En el siguiente flujograma (Ilustración 4.4) se desarrollan estas dos funciones para el caso de estudio, y a continuación se expondrán y explicarán las partes más importantes del código de la aplicación desarrollada.

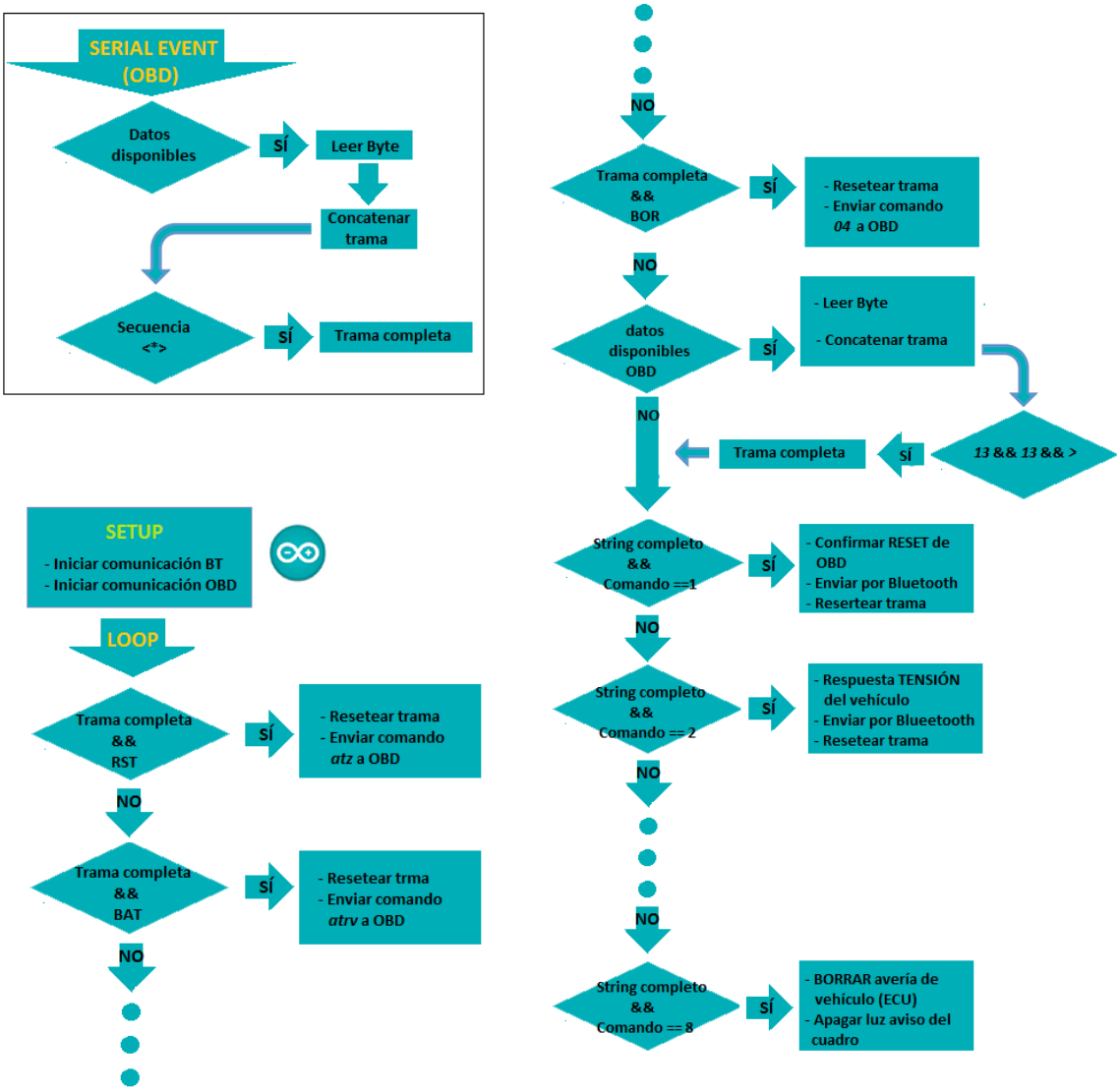


Ilustración 4.4. ***Flujograma Arduino***

En el Listado 4.1. se muestra la inicialización del Sketch que se ha desarrollado para la placa Arduino UNO.

```
byte tipoComando = 0;  //no hacer nada
                       //1 reset
                       //2 leer la bateria
                       //3 leer las rmp sin parar
                       //4 temperatura del refregerante
                       //5 leer km/h
                       //6 averías
                       //7 tipo avería
                       //8 Borrar avería

void setup(){
  inputString.reserve(100);
  inputString1.reserve(100);
  // Comunicación Bluetooth
  Serial.begin(9600);
  // set the data rate for the SoftwareSerial port. UART con OBD
  mySerial.begin(9600);

}
```

Listado 4.1. ***Inicialización Sketch Arduino***

Dicha inicialización consiste en habilitar la UART a 9600 bps (velocidad de funcionamiento del módulo HC-05 Bluetooth), establecer los pines digitales 2 y 3 como un nuevo puerto serial "mySerial" (necesario para habilitar la placa OBD II Uart Hooup Guide ELM327). Los códigos solicitados por AT pueden ser: reset, tensión de la batería (V), velocidad angular (RPM), temperatura del refrigerante (ºC), velocidad lineal (km/h), número y código de averías del vehículo, borrado de avería de la ECU, aunque también es posible programar otros parámetros.

Las acciones fundamentales existentes en el loop() son las siguientes: Procesar el comando recibido por Bluetooth, recibir datos del módulo OBD II Uart Hooup Guide ELM327, y, procesar la respuesta del módulo OBD II Uart Hooup Guide ELM327.

Al ELM327 de la OBD II Uart Hooup Guide se le solicitan la lectura de parámetros de la ECU (Unidad de Mando) del vehículo mediante Bluetooth. En la cadena inputSring1 se almacena los caracteres recibidos por Bluetooth en la placa Arduino utilizando su puerto serial. Si la anterior cadena está completa y coincide con el código asignado, la placa Arduino solicita el parámetro al microprocesador ELM327. En el Listado 4-2 se muestra dicha estructura para, uno de los casos posibles, la lectura de la temperatura de refrigerante del vehículo.

```
void loop(){
  //Procesar el comando recibido por Bluetooth
  if ((stringComplete1) && (inputString1.startsWith("REF"))){  //Tª refrigerante
    inputString1 = "";
    stringComplete1 = false;
    tipoComando = 4;
    mySerial.print("0105");
    mySerial.write(13);
  }
```

Listado 4.2. ***Procesado del comando recibido por Bluetooth***

Para gestionar la selección de datos de la UART de la placa Arduino, recibidos por Bluetooth, se ha utilizado la función "serialEvent()" (ver Listado 4.3). Dicha función se ejecuta cuando se recibe un byte por la UART, almacenando dicho byte en la variable de tipo char denominada "inChar1", que a su vez se van almacenando en una cadena denominada "stingComplete1", hasta recibir el fin de trama ('<' '*' '>'). Si la trama recibida cumple las condiciones se ejecutarán dentro del loop() unas acciones u otras.

```
void serialEvent(){
  while (Serial.available()) {
    // get the new byte:
    char inChar1 = (char)Serial.read();
    // add it to the inputString:
    inputString1 += inChar1;
    // if the incoming character is a newline, set a flag
    // so the main loop can do something about it:
    if ((chAnteriorAnterior1 == '<') && (chAnterior1 == '*') && (inChar1 == '>')) {
      stringComplete1 = true;
    }
    chAnteriorAnterior1 = chAnterior1;
    chAnterior1 = inChar1;
  }
```

Listado 4.3. ***Comandos recibidos en Arduino por Bluetooth***

Para recibir la respuesta del módulo UART-II-OBD se ha utilizado la función "mySerial.read()" (ver Listado 4.4). Dicha función se ejecuta cuando se recibe un byte por la UART y si el "tipoComando" es mayor que cero, almacenando dicho byte en la variable de tipo char denominada "inChar". Dicho byte y los sucesivos se almacenan en otra variable de tipo cadena denominada "inputString" hasta recibir el fin de trama (dos caracteres de tipo retorno de carro).

```
  //Recibir datos del modulo OBD
  while ((mySerial.available()) && (tipoComando > 0)) {
    char inChar = (char)mySerial.read();
    // add it to the inputString:
    inputString += inChar;
    if ((chAnteriorAnterior == 13) && (chAnterior == 13) && (inChar == '>')) {
      stringComplete = true;
    }
    chAnteriorAnterior = chAnterior;
    chAnterior = inChar;
  }
```

Listado 4.4. ***Comandos recibidos del módulo OBD***

A continuación, se procesa la respuesta del OBD II Uart Hooup Guide ELM327 (ver Listado 4.5), agregando la información recibida de la trama en la variable de tipo String "*aux*", seleccionando sólo los datos necesarios y no la trama completa mediante la función "inputString.substring()". Para seleccionar la parte de la trama que corresponde a la respuesta buscada se ha utilizado el programa X-CTU, que permite imprimir las tramas completas (Ilustración 4.5). Como se observa en el caso expuesto, para el requerimiento de la temperatura del refrigerante (ECT o sensor de temperatura del refrigerante), la temperatura del refrigerante es PID (control proporcional integral derivativo) 05 con el modo 01, y se solicita con "0105". La respuesta obtenida es "41 05", que muestra que se

trata de una repuesta a una solicitud de modo PID 1 para 05, mientras que el "4A" es la parte de la trama que contiene el dato realmente buscado. Si se convierte el valor hexadecimal "4A" a decimal el resultado es 74, que representa la temperatura en grados Celsius, con el cero descentralizado para permitir la lectura de temperaturas bajo cero (es decir, valores negativos). Por último, se resta 40 al anterior para convertir el resultado en grados centígrados, resultando 34 ºC, temperatura del refrigerante correspondiente a un motor de un vehículo recién arrancado. La trama termina con un salto de línea (o retorno de carro) expresado por "0D" en hexadecimal ó 13 en decimal, y concluye con el carácter ">" de código ASCII equivalente "3E" en hexadecimal o 62 en código decimal.

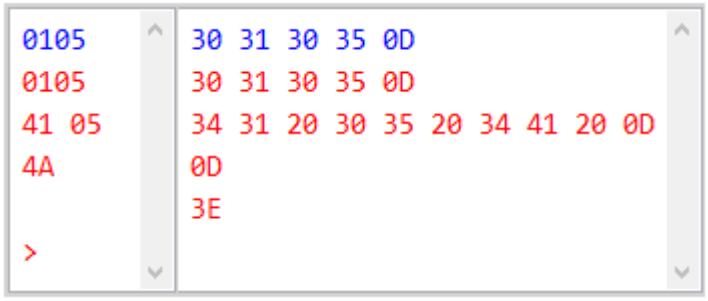


Ilustración 4.5. ***Respuesta del sensor de temperatura obtenida con XCTU***

Para realizar la conversión de la respuesta en Arduino de base hexadecimal, en la que se halla, a base decimal utilizando la función "Strtol()" pero previamente hay que convertir la variable tipo String a tipo char utilizando la función ".toCharArray()" (ver Listado 4.5) Como ha sido comentado con anterioridad, en este caso particular además hay que restarle al valor decimal resultante una constante de 40 para realizar la conversión de escala de temperatura de Celsius a grados centígrados.

Una vez concluido este proceso se envía dicho valor a través de la UART de Arduino, donde está conectado el módulo Bluetooth HC-05 para la impresión de los parámetros en Android (o cualquier dispositivo maestro). Para iniciar la transmisión de datos entre Bluetooth HC-05 y Android Studio (ver Listado 4.5) se imprime un carácter, en este caso particular '&', por el puerto serial, a continuación se imprime el valor del parámetro, la transmisión finaliza cuando se imprime el carácter '~'. Además, se aplica un delay para dar tiempo a que finalice la transmisión.

```
//Procesar la respuesta del OBD
 if (stringComplete && (tipoComando == 4)) {
   String aux = inputString.substring(11, 13);
   Serial.print('&'); //hay que poner & para el comienzo de la trasmisión de los datos,
   para que Android detecte el comienzo del String de datos
   char b[3];
   aux.toCharArray(b,3);
   //Convert the string data to an integer
   ret = (strtol(b, NULL, 16))-40;
   Serial.println(ret);
// clear the string:
   inputString = "";
```

```
    stringComplete = false;
    inputString1 = "";
    //stringComplete1 = false;
    tipoComando = 0;
    Serial.print('~'); //con ~ se da a conocer la finalización del String de datos
    Serial.println();
    delay(10);
   }
```

Listado 4.5. ***Procesar respuesta del módulo OBD***

Con la información anteriormente expuesta se completa el ciclo para la solicitud, procesado, y respuesta de un parámetro, en el caso descrito de la temperatura de refrigerante del motor del vehículo. Para el resto de casos sucede de forma similar.

Un caso particular del código es la detección, cuantificación y borrado de una avería, expuesto en (Listado 4.6 y Listado 4.7). Si se solicita a la placa OBD II Uart Hooup Guide ELM327 el comando avería del automóvil, puede suceder que no exista avería, en cuyo caso se obtendrá que el número de averías es cero; o que existan averías, en cuyo otro caso se imprimirán el número de averías, y el/los códigos de las averías. Finalmente, se podrá proceder al borrado (o reseteo) de la avería de la ECU (Unidad de Mando) del vehículo y apagado de la luz de avería del cuadro de mandos del mismo.

```
else if ((stringComplete1) && (inputString1.startsWith("AVE")))
  {
    inputString1 = "";
    stringComplete1 = false;
    tipoComando = 6;
    mySerial.print("0101"); // Estándar SAE J1979. Modo "01": mostrar los datos actuales
    mySerial.write(13);
  }

if ( i== 1 )  //Si existen averías
  {
    tipoComando = 7;
    mySerial.print("03"); // Estándar SAE J1979. Modo "03": show diagnostic trouble
    codes
    mySerial.write(13);
    i++;
   }

else if ((stringComplete1) && (inputString1.startsWith("BOR"))) //Borrado de averías
  {
    inputString1 = "";
    stringComplete1 = false;
    tipoComando = 8;
    mySerial.print("04"); // Estándar SAE J1979. Modo "04": borrar códigos de problemas
    y valores almacenados
    mySerial.write(13);
  }
```

Listado 4.6. ***Procesado del comando de avería recibido por Bluetooth***

La recepción de datos del módulo OBD II Uart Hooup Guide ELM327, así como la recpeción de solicitud de comando de Bluetooth se realizan igual que en el resto de casos.

En el Listado 4.7 se muestra la estructura empleada para la respuesta del OBD y su transmisión por Bluetooth. Si existen averías se imprimirá el número decimal de averías y los códigos de averías. Los significados de los códigos de averías y su causa se recogen en la

página http://www.teseomotor.com/fallo-obd2/ y en el *Anexo IV* de este Proyecto Fin de Carrera.

```
//Procesar la respuesta del OBD
else if (stringComplete && (tipoComando == 6)) {
    Serial.print('$'); //hay que poner $ para el comienzo de los datos, así Android
    detecta que empieza el String de datos
    String aux = inputString.substring(12, 13);
    Serial.println(aux);//Número de averías
    aux1=aux;
    i=1;
    // clear the string:
    inputString = "";
    stringComplete = false;
    inputString1 = "";
    //stringComplete1 = false;
    tipoComando = 0;

  }

else if (stringComplete && (tipoComando == 7)) {
    String aux ="P"+inputString.substring(9, 14)+ "; P"+inputString.substring(14, 19)+
"; P"+inputString.substring(36, 41);
    if(aux1 != stringTwo){
    Serial.println(inputString);
    Serial.print('+'); //se separan los datos con el +, así es más fácil debuggear
    la información enviada
    Serial.println(aux);
    }
    // clear the string:
    inputString = "";
    stringComplete = false;
    //stringComplete1 = false;
    tipoComando = 0;
    Serial.print('~'); //con ~ se da a conocer la finalización del String de datos
    Serial.println();
    delay(10);
  }
```

Listado 4.7. ***Procesar las respuestas de avería del módulo OBD***

4.4.2 Código Fuente de Android Studio

En esta sección se van a describir los componentes *software* que se han diseñado para la aplicación Android.

Las funcionalidades de la aplicación Android son las siguientes:

- Sincronización del dispositivo Arduino UNO con Android mediante Bluetooth.
- Solicitud por parte de Android a Arduino UNO de los comandos de los parámetros de la placa OBD II Uart Hooup Guide ELM327.
- Visualización gráfica de la conexión Bluetooth.
- Visualización gráfica de los datos procesados.
- Visualización de las tramas de los datos procesados.

A continuación, se describirá la estructura de la aplicación Android para la vista principal, recogida de forma esquemática en la Ilustración 4.6, así como algunos de los métodos utilizados en la aplicación. Finalmente, se describirán las funcionalidades en detalle.

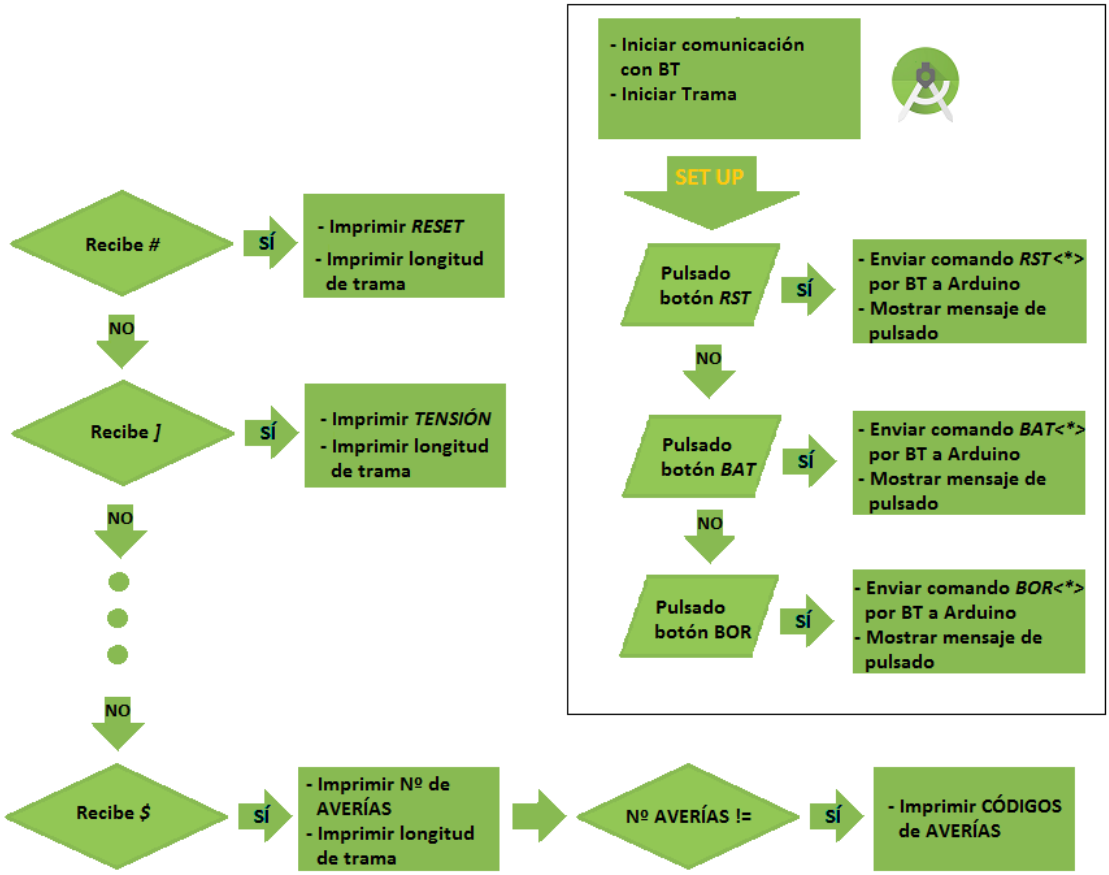


Ilustración 4.6. ***Flujograma Android Activity principal***

En el Listado 4.8. se muestra la inicialización de las variables empleadas, y la inicialización del puerto serie asociado a la comunicación Bluetooth.

```
public class ledControl extends ActionBarActivity {

            Button btnOn, btnOff, btnRPM, btnREF, btnVEL, btnAVE, btnBOR;
            TextView txtArduino,txtString, txtStringLength, sensorView0, sensorView1,
            sensorView2, sensorView3, sensorView4, sensorView5;
            Handler bluetoothIn;

            final int handlerState = 0; //used to identify handler message
            private BluetoothAdapter btAdapter = null;
            private BluetoothSocket btSocket = null;
            private StringBuilder recDataString = new StringBuilder();

            private ConnectedThread mConnectedThread;

            // SPP UUID service - this should work for most devices
            private static final UUID BTMODULEUUID = UUID.fromString("00001101-0000-1000-
            8000-00805F9B34FB");

            // String for MAC address
            private static String address;
```

Listado 4.8. ***Inicialización de variables, y del puerto serial Bluetooth***

En el Listado 4.9 se definen los botones y cuadros de texto de la actividad principal para la pantalla "*Vista de la interfaz desarrollada. Selección de la funcionalidad*" (ver Ilustración 4.9) de la aplicación Android. Se llama al método "*onCreate*" para utilizar la información previa en la creación de la actividad, información que se recibe en el objeto "*savedInstanceState*". También se invoca al método "*setContentView*" (el cual, sólo se puede llamar desde "*onCreate*"), que se utiliza para fijar la vista de la actividad, que es inmutable. Se utiliza normalmente para inicializar la GUI (Interfaz Gráfica de Usuario).

```
@Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);

    setContentView(R.layout.activity_led_control);

    //Link the buttons and textViews to respective views
    btnOn = (Button) findViewById(R.id.button2);
    btnOff = (Button) findViewById(R.id.button3);
    btnRPM = (Button) findViewById(R.id.button6);
    btnREF = (Button) findViewById(R.id.button5);
    btnVEL = (Button) findViewById(R.id.button7);
    btnAVE = (Button) findViewById(R.id.button8);
    btnBOR = (Button) findViewById(R.id.button9);
    txtString = (TextView) findViewById(R.id.txtString);
    txtStringLength = (TextView) findViewById(R.id.testView1);
    sensorView0 = (TextView) findViewById(R.id.sensorView0);
    sensorView1 = (TextView) findViewById(R.id.sensorView1);
    sensorView2 = (TextView) findViewById(R.id.sensorView2);
    sensorView3 = (TextView) findViewById(R.id.sensorView3);
    sensorView4 = (TextView) findViewById(R.id.sensorView4);
    sensorView5 = (TextView) findViewById(R.id.sensorView5);
```

Listado 4.9. *Inicialización de variables, y del puerto serial Bluetooth*

En el Listado 4.10. se muestra el hilo (o *thread*) de transmisión de datos "*handler*" de la aplicación. Al igual que se explicó para el *Sketch* de Arduino, la transmisión de datos por Bluetooth a Android se inicia con la recepción de un carácter prefijado según el parámetro solicitado a la palca OBD II Uart Hooup Guide ELM327, y finaliza con la recepción del carácter "~". La información recibida se muestra por pantalla del Smartphone en dos formatos: un cuadro de texto muestra el valor del parámetro solicitado, y en otro cuadro de texto la trama completa de la cadena recibida. La transmisión de la información se realiza individualmente para cada parámetro solicitado (es decir, la solicitud de un parámetro implica una impresión por pantalla).

```
bluetoothIn = new Handler() {
            public void handleMessage(android.os.Message msg) {
                if (msg.what == handlerState) { //if message is what we want
                    String readMessage = (String) msg.obj; // msg.arg1 = bytes from
                    connect thread
                    recDataString.append(readMessage);//keep appending to string until ~
                    int endOfLineIndex = recDataString.indexOf("~");// determine the
                    end-of-line
                    if (endOfLineIndex > 0) { // make sure there data before ~
                        String dataInPrint = recDataString.substring(0, endOfLineIndex);
                        // extract string
                        txtString.setText("Data Received = " + dataInPrint);
                        int dataLength = dataInPrint.length(); //get length of data
                        received
                        txtStringLength.setText("String Length = " +
                        String.valueOf(dataLength));
```

```
                        if (recDataString.charAt(0) == '#') //if it starts with # we
                        know it is what we are looking for
                        {
                            String sensor0 = recDataString.substring(1, 7); //get sensor
                            value from string between indices 1-5

                            sensorView1.setText(" Reset = " + sensor0 + ";") //update
                            the textviews with sensor values
                        }

                        else if (recDataString.charAt(0) == '¿')
                        {
                            String sensor1 = recDataString.substring(1, 4);//same
                            again...

                            sensorView2.setText(" Bateria = " + sensor1 + "V");
                        }

                        else if (recDataString.charAt(0) == '?')
                        {
                            String sensor2 = recDataString.substring(1, 3);

                            sensorView3.setText(" Velocidad = " + sensor2 + "RPM");
                        }

                        else if (recDataString.charAt(0) == '&')
                        {
                            String sensor3 = recDataString.substring(1, 4);

                            sensorView0.setText(" Temperatura = " + sensor3 + "°C");
                        }

                        else if (recDataString.charAt(0) == '%')
                        {
                            String sensor4 = recDataString.substring(1, 4);

                            sensorView4.setText(" Velocidad = " + sensor4 + "km/h");
                        }

                        else if (recDataString.charAt(0) == '$')
                        {
                            String sensor5 = recDataString.substring(1, 2);
                            //String sensor6 = recDataString.substring(5, 14);

                            sensorView5.setText(" N°Averias = " + sensor5);
                            //sensorView6.setText(sensor6 + ";");
                        }

                        recDataString.delete(0, recDataString.length());
                //clear all string data
                        // strIncom =" ";
                        dataInPrint = " ";
                    }
                }
            }
        };

btAdapter = BluetoothAdapter.getDefaultAdapter(); // get Bluetooth adapter
        checkBTState();
```

Listado 4.10. ***Hilo (Thread) de transmisión de datos Bluetooth desde Arduino***

En el Listado 4.11. se muestra el método "*setup()*" de la aplicación. Este método se llama cuando se pulsa un botón, se utiliza para la solicitud de un comando de la placa OBD II Uart Hooup Guide ELM327 vía Bluetooth. El método implementará toda la lógica de los controles disponibles en la aplicación.

```
// Set up onClick listeners for buttons to send <*>
        btnOff.setOnClickListener(new View.OnClickListener() {
            public void onClick(View v) {
                mConnectedThread.write("BAT<*>");    // Send "BAT<*>" via Bluetooth
                Toast.makeText(getBaseContext(), "Turn on BAT",
```

```
                Toast.LENGTH_SHORT).show();
            }
        });

        btnOn.setOnClickListener(new View.OnClickListener() {
            public void onClick(View v) {
                mConnectedThread.write("RST<*>");    // Send "RST<*>" via Bluetooth
                Toast.makeText(getBaseContext(), "Turn on RST",
                Toast.LENGTH_SHORT).show();
            }
        });

        btnRPM.setOnClickListener(new View.OnClickListener() {
            public void onClick(View v) {
                mConnectedThread.write("RPM<*>");    // Send "RPM<*>" via Bluetooth
                Toast.makeText(getBaseContext(), "Turn on RPM",
                Toast.LENGTH_SHORT).show();
            }
        });

        btnREF.setOnClickListener(new View.OnClickListener() {
            public void onClick(View v) {
                mConnectedThread.write("REF<*>");    // Send "REF<*>" via Bluetooth
                Toast.makeText(getBaseContext(), "Turn on REF",
                Toast.LENGTH_SHORT).show();
            }
        });

        btnVEL.setOnClickListener(new View.OnClickListener() {
            public void onClick(View v) {
                mConnectedThread.write("VEL<*>");    // Send "VEL<*>" via Bluetooth
                Toast.makeText(getBaseContext(), "Turn on VEL",
                Toast.LENGTH_SHORT).show();
            }
        });

        btnAVE.setOnClickListener(new View.OnClickListener() {
            public void onClick(View v) {
                mConnectedThread.write("AVE<*>");    // Send "AVE<*>" via Bluetooth
                Toast.makeText(getBaseContext(), "Turn on AVE",
                Toast.LENGTH_SHORT).show();
            }
        });

        btnBOR.setOnClickListener(new View.OnClickListener() {
            public void onClick(View v) {
                mConnectedThread.write("BOR<*>");    // Send "BOR" via Bluetooth
                Toast.makeText(getBaseContext(), "Turn on BOR",
                Toast.LENGTH_SHORT).show();
            }
        });
    }


    private BluetoothSocket createBluetoothSocket(BluetoothDevice device) throws
    IOException {

        return  device.createRfcommSocketToServiceRecord(BTMODULEUUID);
        //creates secure outgoing connecetion with BT device using UUID
    }
```

Listado 4.11. ***Método setup()***

En el Listado 4.12 se muestra la estructura de la aplicación en caso de fallo de la conexión. Se llama al método "*onCreate*", para utilizar la información previa en la creación de la actividad.

```
    @Override
    public void onPause()
    {
        super.onPause();
        try
        {
```

```
            //Don't leave Bluetooth sockets open when leaving activity
            btSocket.close();
        } catch (IOException e2) {
            //insert code to deal with this
        }
    }

    //Checks that the Android device Bluetooth is available and prompts to be turned on
      if off
    private void checkBTState() {

        if(btAdapter==null) {
            Toast.makeText(getBaseContext(), "Device does not support bluetooth",
            Toast.LENGTH_LONG).show();
        } else {
            if (btAdapter.isEnabled()) {
            } else {
                Intent enableBtIntent = new
                Intent(BluetoothAdapter.ACTION_REQUEST_ENABLE);
                startActivityForResult(enableBtIntent, 1);
            }
        }
    }

    //create new class for connect thread
    private class ConnectedThread extends Thread {
        private final InputStream mmInStream;
        private final OutputStream mmOutStream;

        //creation of the connect thread
        public ConnectedThread(BluetoothSocket socket) {
            InputStream tmpIn = null;
            OutputStream tmpOut = null;

            try {
                //Create I/O streams for connection
                tmpIn = socket.getInputStream();
                tmpOut = socket.getOutputStream();
            } catch (IOException e) { }

            mmInStream = tmpIn;
            mmOutStream = tmpOut;
        }


        public void run() {
            byte[] buffer = new byte[256];
            int bytes;

            // Keep looping to listen for received messages
            while (true) {
                try {
                    bytes = mmInStream.read(buffer);//read bytes from input buffer
                    String readMessage = new String(buffer, 0, bytes);
                    // Send the obtained bytes to the UI Activity via handler
                    bluetoothIn.obtainMessage(handlerState, bytes, -1,
                    readMessage).sendToTarget();
                } catch (IOException e) {
                    break;
                }
            }
        }
        //write method
        public void write(String input) {
            byte[] msgBuffer = input.getBytes(); //converts entered String into bytes
            try {
                mmOutStream.write(msgBuffer);//write bytes over BT connection via
                outstream
            } catch (IOException e) {
                //if you cannot write, close the application
                Toast.makeText(getBaseContext(), "Connection Failure",
                Toast.LENGTH_LONG).show();
                finish();

            }
        }
```

```
    }
```

Listado 4.12. ***Estructura de aplicación para el fallo de la conexión***

A continuación, en el Listado 4.13 se muestra el archivo ".xml" de la vista principal de la actividad, y los archivos fundamentales del ".xml" que corresponden a cada vista.

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools" android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    android:paddingBottom="@dimen/activity_vertical_margin"
    tools:context="com.led.led.ledControl"
    android:background="@drawable/contador1"
    android:orientation="vertical"
    android:weightSum="10">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Diagnosis Control"
        android:id="@+id/textView2"
        android:layout_alignParentTop="true"
        android:layout_alignParentLeft="true"
        android:layout_alignParentStart="true"
        android:textColor="#750A0A"/>

    <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
        android:orientation="horizontal"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:layout_below="@+id/textView2">

      <Button
         android:layout_width="wrap_content"
         android:layout_height="wrap_content"
         android:text="1.RST"
         android:id="@+id/button2"
         android:layout_marginTop="10dp"
         android:layout_alignParentLeft="true"
         android:layout_alignParentStart="true"/>
…
</LinearLayout>
```

Listado 4.13. ***Archivo .xml Vista princial de la Actividad***

Por último, se describirá la estructura de la aplicación Android para la pantalla "*Vista de la interfaz desarrollada. Selección dispositivo del Bluetooth"* de la aplicación (ver Ilustración 4.8), así como algunos de los métodos utilizados en la aplicación. Finalmente, se describirá las funcionalidades en detalle.

En el Listado 4.14. se observa la clase creada por defecto para una aplicación Android extiende de un "*Activity*" (Actividad), en este caso extiende de "*ActionBarActivity*". Una de las acciones llevadas a cabo dentro del "*onCreate*" es enlazar los controles que forman la vista con los controles de la actividad, para poder darle funcionalidad posteriormente a ese control.

```
public class DeviceList extends ActionBarActivity
{
    Button btnVinculados;
    ListView listaDispositivos;
```

```
private BluetoothAdapter myBluetooth = null;
private Set<BluetoothDevice> dispVinculados;
public static String EXTRA_ADDRESS = "device_address";

@Override
protected void onCreate(Bundle savedInstanceState)
{
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_device_list);

    //Se declaran los componenetes ralcionandolos con los del layout
    btnVinculados = (Button)findViewById(R.id.button);
    listaDispositivos = (ListView)findViewById(R.id.listView);

    //Comprobamos que el dispositivo tiene bluetooth
    myBluetooth = BluetoothAdapter.getDefaultAdapter();

    if(myBluetooth == null)
    {
        //Se muestra un mensaje, indicando al usuario que no tiene conexión
          bluetooth disponible
        Toast.makeText(getApplicationContext(), "Bluetooth no disponible",
        Toast.LENGTH_LONG).show();

        //Se finaliza la aplicación
        finish();
    }
    else if(!myBluetooth.isEnabled())
    {
            //Se pregunta al usuario si desea encender el bluetooth
            Intent turnBTon = new Intent(BluetoothAdapter.ACTION_REQUEST_ENABLE);
            startActivityForResult(turnBTon,1);
    }

    btnVinculados.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v)
        {
            listaDispositivosvinculados();
        }
    });

}

private void listaDispositivosvinculados()
{
    dispVinculados = myBluetooth.getBondedDevices();
    ArrayList list = new ArrayList();

    if (dispVinculados.size()>0)
    {
        for(BluetoothDevice bt : dispVinculados)
        {
            list.add(bt.getName() + "\n" + bt.getAddress()); //Se obtienen los
            nombres y direcciones MAC de los disp. vinculados
        }
    }
    else
    {
        Toast.makeText(getApplicationContext(), "No se han encontrado dispositivos
        vinculados", Toast.LENGTH_LONG).show();
    }

    final ArrayAdapter adapter = new
    ArrayAdapter(this,android.R.layout.simple_list_item_1, list);
    listaDispositivos.setAdapter(adapter);
    listaDispositivos.setOnItemClickListener(myListClickListener);

}

private AdapterView.OnItemClickListener myListClickListener = new
AdapterView.OnItemClickListener()
{
    public void onItemClick (AdapterView<?> av, View v, int arg2, long arg3)
    {
        // Get the device MAC address, the last 17 chars in the View
        String info = ((TextView) v).getText().toString();
```

```
            String address = info.substring(info.length() - 17);

            // Make an intent to start next activity.
            Intent i = new Intent(DeviceList.this, ledControl.class);

            //Change the activity.
            i.putExtra(EXTRA_ADDRESS, address); //this will be received at class
            Activity
            startActivity(i);
        }
    };


    @Override
    public boolean onCreateOptionsMenu(Menu menu)
    {
        getMenuInflater().inflate(R.menu.menu_device_list, menu);
        return true;
    }

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        // Handle action bar item clicks here. The action bar will
        // automatically handle clicks on the Home/Up button, so long
        // as you specify a parent activity in AndroidManifest.xml.
        int id = item.getItemId();
        //noinspection SimplifiableIfStatement
        if (id == R.id.action_settings) {
            return true;
        }
        return super.onOptionsItemSelected(item);
    }
}
```

Listado 4.14. ***Clase de la Actividad de conexión Bluetooth***

Por último, se muestra el archivo ".xml" para la pantalla de la "*Vista de la interfaz desarrollada. Selección dispositivo del Bluetooth"* de la actividad, y los archivos fundamentales del ".xml" que corresponden a cada vista.

Como se observa en el Listado 4.15. en esta ocasión se ha empleado un *RetativeLayout* para la construcción de la vista. Además, se emplea una Lista de Vistas (*ListView*).

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools" android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    android:background="@drawable/bluetooth1"
    android:paddingBottom="@dimen/activity_vertical_margin" tools:context=".DeviceList">
…
<ListView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/listView"
        android:layout_centerHorizontal="true"
        android:layout_above="@+id/button"
        android:layout_below="@+id/textView" />
</RelativeLayout>
```

Listado 4.15. ***Método Control Vista de conexión de Bluetooth***

En el siguiente flujograma (Ilustración 4.7) se describe sistemáticamente el funcionamiento de la Activity para la solicitud de conexión Bluetooth.

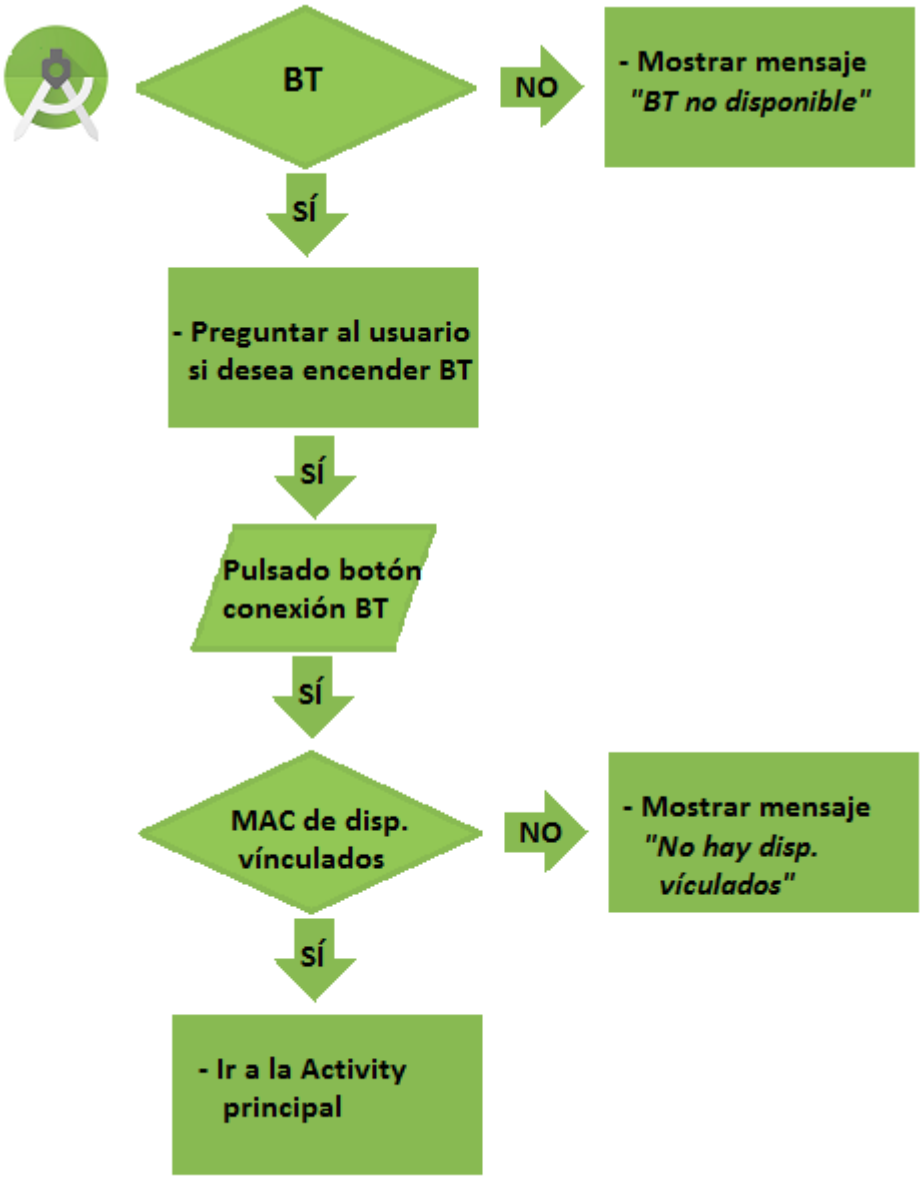


Ilustración 4.7. ***Flujograma Android Activity Bluetooth***

4.5 Descripción del funcionamiento

La aplicación Android consta de dos vistas referidas en el apartado anterior, que se detallarán a continuación. La primera vista muestra la conexión Bluetooth, la segunda vista sirve para la solicitud de los comandos de los parámetros e impresión se sus respuestas.

La primera vista, denominada "*Vista de la interfaz desarrollada. Selección dispositivo del Bluetooth*" permite establecer la conexión Bluetooth, solicitar las direcciones MAC de los dispositivos vinculados y emparejarlo con uno de los mostrados. Es decir, integra la acción de sincronización de Arduino UNO con el Smartphone Android mediante Bluetooth.

Ilustración 4.8: ***Vista de la interfaz desarrollada. Selección dispositivo del Bluetooth***

Como se observa en la Ilustración 4.8, existe un botón denominado *"Dispositivos vinculados"* que busca dispositivos esclavos conectados. Además, esta primera vista incluye un *TextView* que muestra un mensaje al pulsar el botón anterior, y un control para visualizar texto denominado *ListView,* que permite seleccionar el dispositivo a emparejar por Bluetooth. Cuando la conexión falla se activa un aviso "*Toast*" que informa al usuario.

Si el Bluetooth del dispositivo Smartphone está apagado, aparecerá un aviso que permitirá encenderlo desde la aplicación.

En la Ilustración 4.9. se muestra la segunda vista. En la segunda vista, denominada "*Vista de la interfaz desarrollada. Selección de la funcionalidad*" se integran las siguientes acciones, solicitud de los parámetros reset, tensión de la batería (V), velocidad angular (RPM), temperatura del refrigerante (ºC), velocidad lineal (km/h), número y código de averías del vehículo, y el borrado de avería de la ECU.

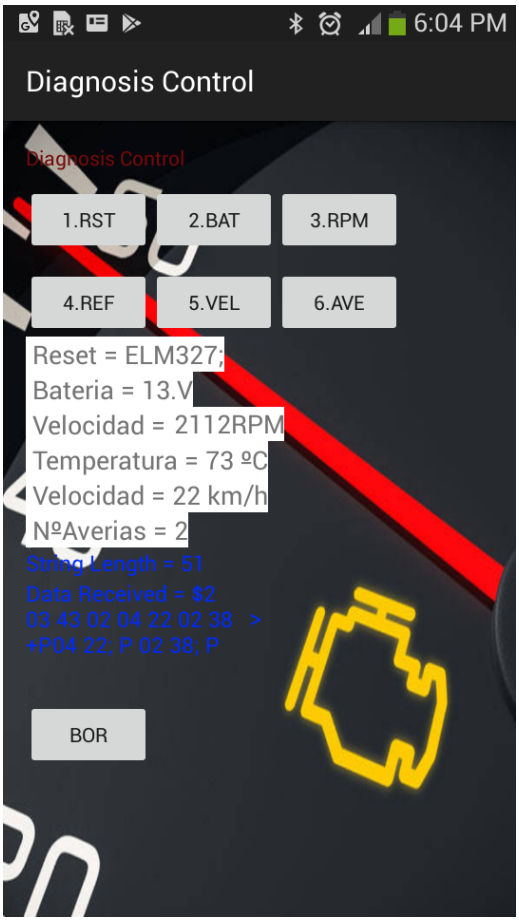


Ilustración 4.9: ***Vista de la interfaz desarrollada. Selección de la funcionalidad***

Tal como se observa en la Ilustración 4.9, la actividad consta de siete botones denominados "RST", "BAT", "RPM", "REF", "VEL", "AVE" y "BOR" para las acciones comentadas anteriormente, además de ocho controles para visualizar texto.

Al pulsar cualquiera de los botones mencionados se muestra un aviso "*Toast*" para informar del usuario.

Toda la información expuesta en el presente capítulo, de este proyecto fin de carrera, se encuentra ampliada en los anexos I, II, III, IV, V y VI.

Capítulo 5

Conclusiones y Trabajos futuros

5.1 Conclusiones

En este proyecto se ha abordado diseño y desarrollo un sistema de diagnóstico de vehículos basado en módulos empotrados de bajo coste y en dispositivos móviles inteligentes, aplicación que podría resultar de gran utilidad para los propietarios de vehículo por ser sencilla de utilizar y eficiente.

Se ha realizado un estudio para escoger los dispositivos empotrados y la plataforma más adecuada. Para los dispositivos empotrados, se ha tenido en cuenta el coste, su facilidad de integración, su universalidad, sus prestaciones y su grado de comercialización en la actualidad. Se han seleccionado el módulo OBD II Uart Hookup Guide ELM327, módulo Bluetooth HC-05, y la placa Arduino. En el caso de la plataforma se han tenido en cuenta el coste, la importancia de la plataforma y su potencial, siendo Android la elegida por su gran difusión en la actualidad.

Se ha llevado a cabo un caso de estudio en el que se ha diseñado y ejecutado una aplicación de diagnóstico de averías de automóvil, que es capaz de leer parámetros del motor, como tensión de la batería, velocidad angular (RMP), temperatura de refrigerante, velocidad lineal (km/h), número de averías, y código de averías. Además, permite resetear el microchip ELM327 que integra la placa OBD II Uart Hookup, encargado de realizar las lecturas, y también el borrado de averías de la Unidad de Mando (ECU) y apagado de las luces de

averías del cuadro de mandos del vehículo. Con este fin, se ha implementado el hardware necesario mediante la integración de sensores a los dispositivos empotrados y se ha obtenido el software mediante el desarrollo de los archivos pertinentes utilizando Android Studio con leguaje *Java* y XML, y Arduino con lenguaje *C++*. En el caso de Arduino, se han creado todos los Sketches necesarios, y para OBD II Uart Hookup, se ha utilizado el lenguaje *C++* para implementar todos los componentes software necesarios. Finalmente, se ha diseñado una aplicación Android que aporta una interfaz gráfica al sistema utilizado Android Studio. Con ella se pueden llevar a cabo la visualización de parámetros del automóvil, tales como tensión de la batería (V), velocidad angular (RPM), temperatura del refrigerante (ºC), velocidad lineal (km/h), y número y código de averías. Además, permite el borrado de averías de la Unidad de Mando del vehículo y apagado de luces de averías del cuadro de mandos del vehículo. Se han empleado para para tal fin los lenguajes de etiquetas XML, y el lenguaje Java. La conexión del Smartphone inteligente (también pude utilizarse otros dispositivos como PC o tabletas) con el módulo de diagnóstico se realiza por Bluetooth, módulo que hace muy practicable el emparejamiento de infinidad de dispositivos comerciales.

Finalmente, y a modo de resumen, con este estudio se ha conseguido la realización de una aplicación económica, eficiente y de aplicación sencilla, que permite realizar diagnósticos de vehículos, lectura de parámetros del motor y borrado de averías, según los protocolos normalizados y universales ISO y SAE, con un sinfín de dispositivos de uso comercial como: móviles inteligente Smartphone, PC y tabletas. La creciente implantación de la electrónica en el automóvil da muestra del verdadero potencial de esta tecnología.

5.2 Trabajos Futuros

Como posible trabajo futuro, se puede crear una aplicación con mayor número de funciones mediante la implementación de otros parámetros que soliciten información complementaria al vehículo. Se podría manipular los parámetros de la Unidad de Mando para controlar el vehículo desde la aplicación. Sería muy interesante el poder manipular la codificación de la Unidad de Mando (ECU), para modificar limitaciones. Se pueden aplicar acciones correctivas sobre los parámetros del vehículo analizado, y no únicamente centrarse en la lectura y borrado de los mismos. Se puede complementar la aplicación desarrollada con Android Studio. Se pueden cargar los códigos y las instrucciones de reparación de averías, para que el usuario sea capaz de realizar la reparación de su vehículo, por ejemplo, empleando realidad virtual.

Anexo I

Sketch Arduino

A continuación, se muestra (ver Listado I.1) el *Sketch* Arduino integro, de la aplicación desarrollada:

```
SoftwareSerial mySerial(2,3); // RX, TX

String inputString = "";         // a string to hold incoming data
String inputString1 = "";         // a string to hold incoming data
boolean stringComplete = false;  // whether the string is complete
boolean stringComplete1 = false;  // whether the string is complete
char chAnterior, chAnterior1;
char chAnteriorAnterior, chAnteriorAnterior1;
long ret;
int i;
String aux1;
String stringTwo= "0";


byte tipoComando = 0;  //no hacer nada
                       //1 reset
                       //2 leer la bateria
                       //3 leer las rmp sin parar
                       //4 temperatura del refregerante
                       //5 leer km/h
                       //6 averías
                       //7 tipo avería
                       //8 Borrar avería

void setup()
{
  inputString.reserve(100);
```

```
  inputString1.reserve(100);


  // Comunicación Bluetooth
  Serial.begin(9600);

  // set the data rate for the SoftwareSerial port. UART con OBD
  mySerial.begin(9600);

}

////////////////////////////////////////////
////////////////////////////////////////////
void loop()
{
  //Procesar el comando recibido por Bluetooth
  if ((stringComplete1) && (inputString1.startsWith("RST")))  // Command is to reset
  {
    inputString1 = "";
    stringComplete1 = false;
    tipoComando = 1;
    mySerial.print("atz");
    mySerial.write(13);

  }
  else if ((stringComplete1) && (inputString1.startsWith("BAT")))  // Battery read
  {
    inputString1 = "";
    stringComplete1 = false;
    tipoComando = 2;
    mySerial.print("atrv");
    mySerial.write(13);
  }
  else if ((stringComplete1) && (inputString1.startsWith("RPM")))  // RPM Reading
  {
    inputString1 = "";
    stringComplete1 = false;
    tipoComando = 3;
    mySerial.print("010c");
    mySerial.write(13);
  }
  else if ((stringComplete1) && (inputString1.startsWith("REF")))    //  Refrigerator
temperature Reading
  {
    inputString1 = "";
    stringComplete1 = false;
    tipoComando = 4;
    mySerial.print("0105");
    mySerial.write(13);
  }
 else if ((stringComplete1) && (inputString1.startsWith("VEL")))  // km/h Reading
  {
    inputString1 = "";
    stringComplete1 = false;
    tipoComando = 5;
    mySerial.print("010D");
    mySerial.write(13);
  }
 else if ((stringComplete1) && (inputString1.startsWith("AVE")))
```

```
  {
    inputString1 = "";
    stringComplete1 = false;
    tipoComando = 6;
    mySerial.print("0101");  //  Estándar  SAE  J1979.  Modo  "01":  mostrar  los  datos
actuales
    mySerial.write(13);
  }

 else if ((stringComplete1) && (inputString1.startsWith("BOR")))
  {
    inputString1 = "";
    stringComplete1 = false;
    tipoComando = 8;
    mySerial.print("04"); // Estándar SAE J1979. Modo "04": borrar códigos de problemas
y valores almacenados
    mySerial.write(13);
  }

else if ( i== 1 )  //(inputString1.startsWith("AUX"))
  {
    tipoComando = 7;
    mySerial.print("03");  //  Estándar  SAE  J1979.  Modo  "03":  show  diagnostic  trouble
codes
    mySerial.write(13);
    i++;
   }


  //Recibir datos del modulo OBD
  while ((mySerial.available()) && (tipoComando > 0)) {
    char inChar = (char)mySerial.read();
    // add it to the inputString:
    inputString += inChar;
    if ((chAnteriorAnterior == 13) && (chAnterior == 13) && (inChar == '>')) {
      stringComplete = true;
    }
    chAnteriorAnterior = chAnterior;
    chAnterior = inChar;
  }


  //Procesar la respuesta del OBD
  if (stringComplete && (tipoComando == 1)) {
    String aux = inputString.substring(6, 12);
    Serial.print('#'); // el carácter # indica el comienzo de transmisión de datos por
BT, para que Android identifique el inicio del String
    Serial.println(aux);
    // clear the string:
    inputString = "";
    stringComplete = false;
    inputString1 = "";
    //stringComplete1 = false;
    tipoComando = 0;
    Serial.print('~'); //con esto se da a conocer la finalización del String de datos
    Serial.println();
    delay(10);
  }
  else if (stringComplete && (tipoComando == 2)) {
```

```
    Serial.print(']'); // el carácter ] indica el comienzo de transmisión de datos por
BT, para que Android identifique el inicio del String
    String aux = inputString.substring(5, 9);
    Serial.println(aux);
    // clear the string:
    inputString = "";
    stringComplete = false;
    inputString1 = "";
    //stringComplete1 = false;
    tipoComando = 0;
    Serial.print('~'); //con esto se da a conocer la finalización del String de datos
    Serial.println();
    delay(10);
  }
  else if (stringComplete && (tipoComando == 3)) {
    String aux = inputString.substring(11, 18);
    Serial.print('?'); // el carácter ? indica el comienzo de transmisión de datos por
BT, para que Android identifique el inicio del String
    char b[8];
    aux.toCharArray(b,8);
    //Convert the string data to an integer
    ret = (strtol(b, NULL, 16))*66;
    Serial.println(ret);
    // clear the string:
    inputString = "";
    stringComplete = false;
    inputString1 = "";
    //stringComplete1 = false;
    tipoComando = 0;
    Serial.print('~'); //con esto se da a conocer la finalización del String de datos
    Serial.println();
    delay(10);
  }
else if (stringComplete && (tipoComando == 4)) {
   String aux = inputString.substring(11, 13);
   Serial.print('&'); // el carácter & indica el comienzo de transmisión de datos por
BT, para que Android identifique el inicio del String
   char b[3];
   aux.toCharArray(b,3);
   //Convert the string data to an integer
   ret = (strtol(b, NULL, 16))-40;
   Serial.println(ret);
   inputString = "";
   stringComplete = false;
   inputString1 = "";
   //stringComplete1 = false;
   tipoComando = 0;
   Serial.print('~'); //con esto se da a conocer la finalización del String de datos
   Serial.println();
   delay(10);
  }
else if (stringComplete && (tipoComando == 5)) {
    String aux = inputString.substring(10, 14);
    Serial.print('%'); // el carácter % indica el comienzo de transmisión de datos por
BT, para que Android identifique el inicio del String
    char b[8];
    aux.toCharArray(b,8);
    ret = (strtol(b, NULL, 16));
    Serial.println(ret);
```

```
    // clear the string:
    inputString = "";
    stringComplete = false;
    inputString1 = "";
    //stringComplete1 = false;
    tipoComando = 0;
    Serial.print('~'); //con esto se da a conocer la finalización del String de datos
    Serial.println();
    delay(10);
  }

else if (stringComplete && (tipoComando == 6)) {
    Serial.print('$'); // el carácter $ indica el comienzo de transmisión de datos por
BT, para que Android identifique el inicio del String
    String aux = inputString.substring(12, 13);
    Serial.println(aux);//Número de averías.
    aux1=aux;
    i=1;
    // clear the string:
    inputString = "";
    stringComplete = false;
    inputString1 = "";
    //stringComplete1 = false;
    tipoComando = 0;
    //Serial.print('+'); //se separan los datos con el +, así se facilita el debuggear
de la información que enviada
  }

else if (stringComplete && (tipoComando == 7)) {
    String aux ="P"+inputString.substring(9, 14)+ "; P"+inputString.substring(14, 20)+
"; P"+inputString.substring(36, 41);
    if(aux1 != stringTwo){
    Serial.println(inputString);
    Serial.print('+'); //se separan los datos con el +, así se facilita el debuggear de
la información que enviada
    Serial.println(aux);
    //aux1 == 0;
    }
    // clear the string:
    inputString = "";
    stringComplete = false;
    //stringComplete1 = false;
    tipoComando = 0;
    Serial.print('~'); //con esto se da a conocer la finalización del String de datos
    Serial.println();
    delay(10);
  }
}

////////////////////////////////////////////////
////////**********************/////////////////////
////////////////////////////////////////////////
void serialEvent(){
  while (Serial.available()) {
    // get the new byte:
    char inChar1 = (char)Serial.read();
    // add it to the inputString:
    inputString1 += inChar1;
    // if the incoming character is a newline, set a flag
```

```
    // so the main loop can do something about it:
    if ((chAnteriorAnterior1 == '<') && (chAnterior1 == '*') && (inChar1 == '>')) {
      stringComplete1 = true;
    }
    chAnteriorAnterior1 = chAnterior1;
    chAnterior1 = inChar1;
  }
}
```

Listado I.1. ***Sketch Arduino***

Anexo II

Interpretación y comentario del significado de las tramas del UART-II-OBD to ELM327 obtenidas con el programa XCTU

Las tramas estudiadas a continuación se han obtenido utilizando el programa XCTU. Las respuestas resultantes están en código ASCII en la parte izquierda de las ilustraciones, y en código hexadecimal en la parte derecha.

I. Reset placa OBD

El requerimiento de Reset para la placa OBD II Uart Hookup se solicita con el comando "atz" (ver Ilustración II.1), que es conveniente cuando se inicia la toma de medidas. La placa devuelve "ELM327 v1.3a" a modo de confirmación.

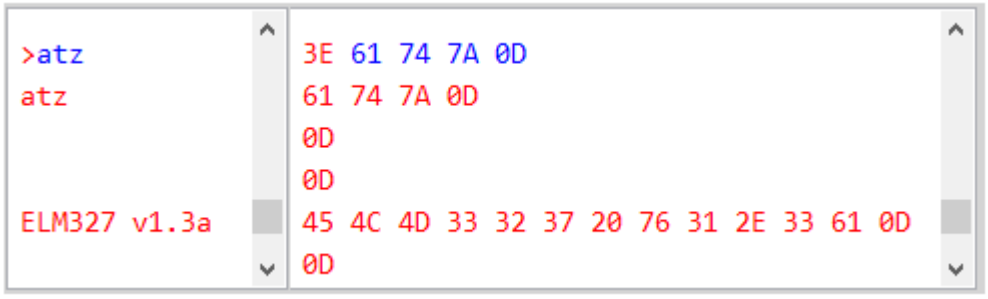


Ilustración II.1. ***Código de Reset placa OBD***

II. Tensión de la batería

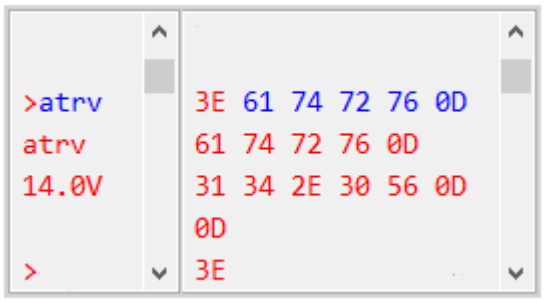


Ilustración II.2. ***Código de Tensión de la batería***

El requerimiento de la tensión de la batería del vehículo se solicita con "atrv" (ver Ilustración II.2). La respuesta obtenida es la tensión 14.0V para el caso de estudio, a modo de ejemplo (ver Tabla II.1):

ASCII	DECIMAL	HEXADECIMAL
1	49	31
4	52	34
.	46	2E
0	48	30
V	86	56
Salto de carro	13	0D
>	62	3E

Tabla II.1. ***Conversión de datos***

La trama termina con un salto de línea (o retorno de carro) expresado por "0D" en hexadecimal o 13 en decimal, y concluye con el carácter ">" de código ASCII equivalente "3E" en hexadecimal o 62 en código decimal.

III. Velocidad angular (RPM)

El requerimiento de velocidad angular (rpm), es el modo de PID 0C 01, que se solicita con el indicador "01 0C" (ver Ilustración II.3).
Con el motor parado, la respuesta obtenida para el caso de estudio es "41 0C 00 00". El valor devuelto (00 00) es cero.

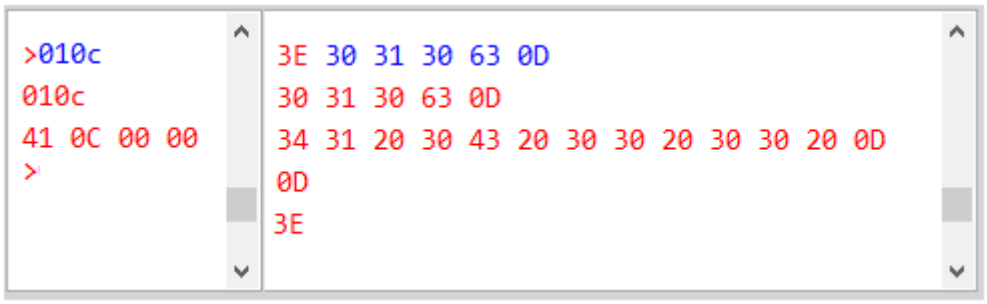


Ilustración II.3. ***Código de Velocidad angular (RPM) con el motor parado***

Si el motor está en marcha, la respuesta para el caso de estudio es "41 0C 18 0B" (ver Ilustración II.4).

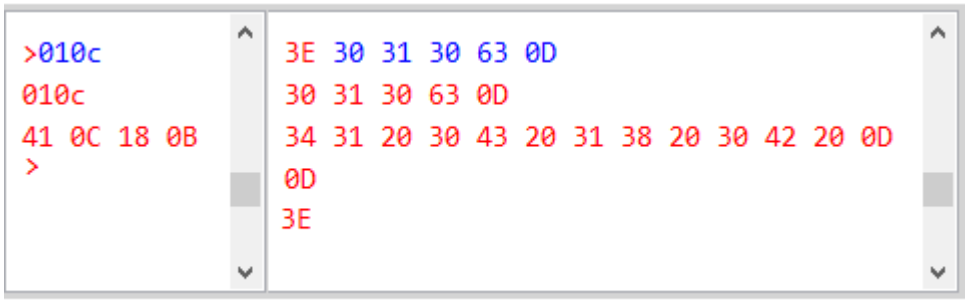


Ilustración II.4. ***Código de Velocidad angular (RPM) con el motor en marcha***

El valor devuelto "18 0B" es en realidad un byte de dos número hexadecimal que se debe convertir a un valor decimal para ser útil. Realizando la conversión se obtiene un valor de 6.155, que es muy alto de revoluciones del motor. Esto es así porque se envían rpm incrementadas por 4 (debido al giro del cigüeñal por ser un motor de 4 Tiempos).
Para obtener la velocidad real del motor, es necesrio realizar una regresión lineal con la que se obitiene al dividir un valor de 1538 rpm.

IV. Temperatura del refrigerante

Para el requerimiento de la temperatura del refrigerante (ECT o sensor de temperatura del refrigerante), la temperatura del refrigerante es PID (control proporcional integral derivativo) 05 con el modo 01, y se solicita con "0105" (ver Ilustración II.5). La respuesta obtenida es "41 05", que muestra que se trata de una repuesta a una solicitud de modo PID 1 para 05, mientras que el "4A" es la parte de la trama que contiene el dato realmente buscado. Si se convierte el valor hexadecimal "4A" a decimal el resultado es 74, que representa la temperatura en grados Celsius, con el cero descentralizado para permitir la lectura de temperaturas bajo cero (es decir, valores negativos). Por último, se resta 40, al anterior valor en grados Celsius, para convertir en grados centígrados, resultando 34ºC, temperatura del refrigerante correspondiente a un motor de un vehículo recién arrancado. La trama termina con un salto de línea (o retorno de carro) expresado por "0D" en hexadecimal o 13 en decimal, y concluye con el carácter ">" de código ASCII equivalente "3E" en hexadecimal o 62 en código decimal.

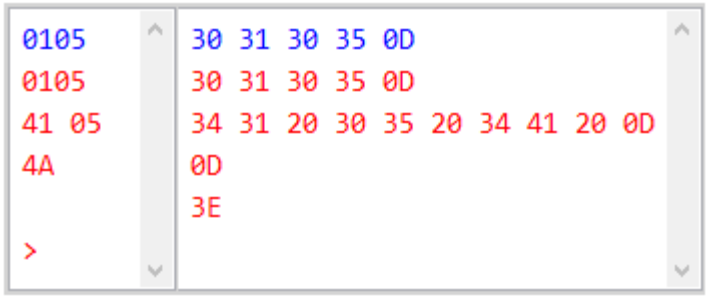


Ilustración II.5. ***Código de Temperatura del Refrigerante***

V. Velocidad lineal (km/h)

Para el requerimiento de la velocidad lineal, la velocidad es PID (control proporcional integral derivativo) 01 con el modo 0D, y se solicita con "010D" (ver Ilustración II.6). La respuesta obtenida es "41 0D 00", que muestra que se trata de una repuesta a una solicitud de modo PID 1 para "0D", mientras que el "00" es la parte de la trama que contiene el dato realmente buscado, correspondiente a la velocidad 0 km/h por estar el vehículo estático en el momento de la medición . La trama termina con un salto de línea (o retorno de carro) expresado por "0D" en hexadecimal o 13 en decimal, y concluye con el carácter ">" de código ASCII equivalente "3E" en hexadecimal o 62 en código decimal.

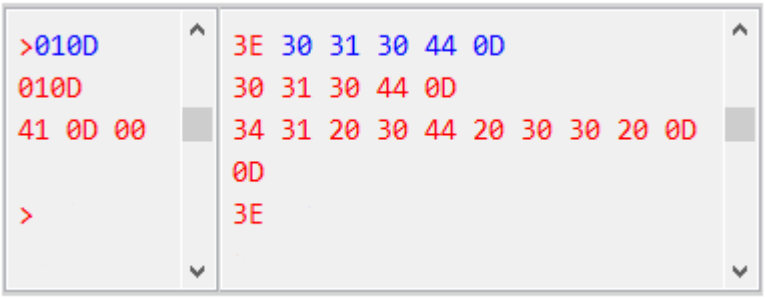


Ilustración II.6. ***Código de Velocidad lineal (km/h)***

VI. Determinación del número y código de averías

Para conocer el código de avería se requiere un modo de 03, pero primero debe cuantificar cuántos códigos de averías se almacenan en la actualidad en la ECU (Unidad de Mando del vehículo). Esto se consigue con un modo 01 PID 01 solicitado (ver Ilustración II.7) de la siguiente forma "01 01", a lo que una respuesta típica podría ser "41 01 81 07 61 61".

El 41 01 significa una respuesta a la solicitud, y el byte siguiente de datos (81) es el número de problemas actuales. Es evidente que no habrá 81 (hex) ó 129 (Decimal) códigos de error presentes si el vehículo está opertativo. De hecho, este byte cumple una doble función, el bit más significativo se utiliza para indicar que la luz de avería o mal funcionamiento del cuadro de mandos del vehículo (MIL, o "Check Engine Luz ') se halla encendida e indica como mínimo una avería, mientras que el otro 7 bits de este byte proporcionan el número real de almacenado los códigos de problemas.

Con el fin de calcular el número de códigos almacenados cuando el MIL está encendida, sólo hay que restar a 128 (ó 80 hex), el 129 (ó 81 hex) en este caso de estudio, obteniéndose que el número de códigos de averías del vehículo es uno. La respuesta anterior, entonces indica que hay un código de averías almacenado para la luz de averías del cuadro de mandos del vehículo o MIL.

El resto de los bytes en la respuesta proporciona información sobre los tipos de pruebas del caso de estudio en particular (ver el J1979 documento para obtener más información).

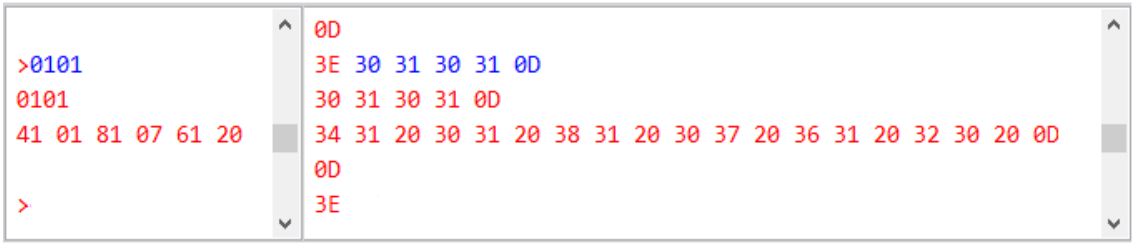


Ilustración II.7. ***Código de Número de averías***

En el caso de no haber ningún tipo de avería y estar la luz de avería del cuadro de mandos del vehículo apagada la respuesta sería la sigueinte "41 01 00 00 00 00" (ver Ilustración II.8).

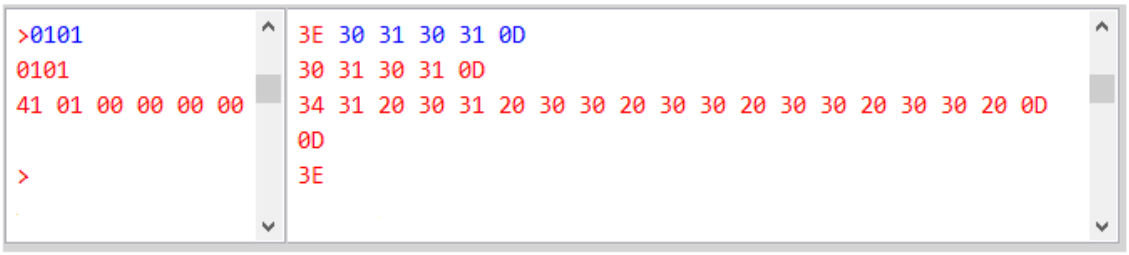


Ilustración II.8. ***Código para número de averías cero***

El 41 01 significa una respuesta a la solicitud como ya se ha comentado anteriormente, y el byte siguiente de datos (00) indica que la luz de averías del cuadro de mandos del vehículo se halla apagada.

Una vez determinado el número de códigos almacenados, el siguiente paso es solicitar los códigos de problemas reales con una petición de modo "03" (no hay PID porque no es necesario). Una respuesta a esto podría ser "43 01 02 38 00 00 00".

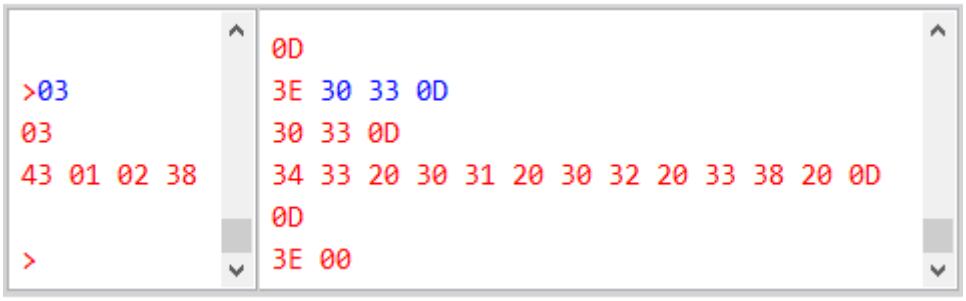


Ilustración II.9. ***Código de Avería***

El '43' en la respuesta anterior simplemente indica que esta es una respuesta a un modo de 03 solicitud. Los otros 6 bytes en la respuesta tienen que ser leído a pares para mostrar los códigos de problemas (lo anterior habría interpretado como 0102, 3800 y 0000).

Tomando el ejemplo de código de avería anterior (0102), el primer dígito (0) entonces sería reemplazado por P0 y el 0102, que contiene el código de avería, que se convertiría en P0102, que es el código de avería de "Caudalímetro, Señal baja " (Ver *Anexo VII Códigos de averías*).

En caso de que el número de averías sea cero el resultado obtenido sería “43 00 00 00 00 00 00” (ver Ilustración II.10).

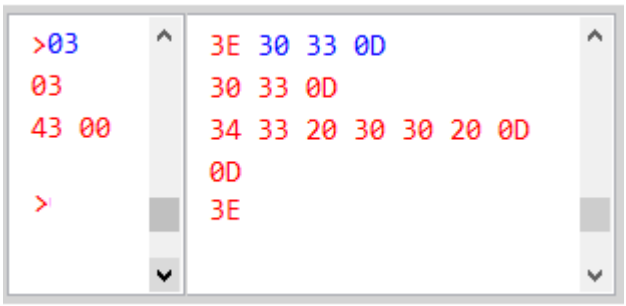


Ilustración II.10. ***Código de Averías con MIL apagada***

VII. Estándar SAE J1979. Modo "04": borrar códigos de problemas y valores almacenados

El requerimiento del borrado o restablecimiento de códigos de averías se solicita con un modo “04” (ver Ilustración II.11). La solicitud realiza:

- Restablecer el número de códigos de problemas
- Borrar los códigos de diagnóstico
- Borrar todos los datos congelados y almacenados del cuadro
- Borrar el DTC que inició la imagen congelada
- Borrar todos los datos de prueba del sensor de averiado

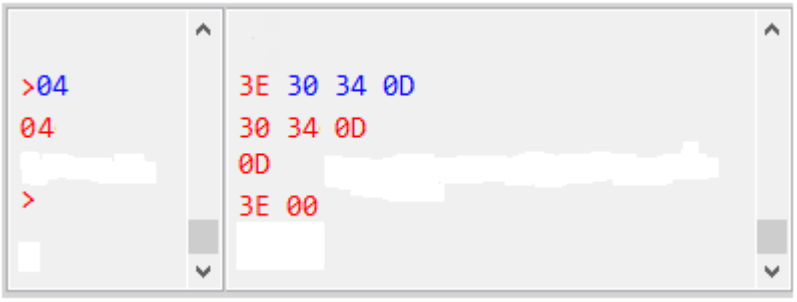


Ilustración II.11. ***Código de Borrado de averías***

No hay trama de respuesta a la solicitud de este modo “04”, la respuesta se manifiesta mediante el apagado de la luz de avería del cuadro de mandos del vehículo, y borrado de las averías de la Unidad de Mando del vehículo.

Anexo III

Android Studio, Código Java Activity Principal

A continuación se muestra (ver Listado III.1) el código Java integro de Android Studio para la Activity Principal, de la aplicación desarrollada:

```
package com.led.led;

import android.bluetooth.BluetoothAdapter;
import android.bluetooth.BluetoothDevice;
import android.bluetooth.BluetoothSocket;
import android.content.Intent;
import android.os.Bundle;
import android.os.Handler;
import android.support.v7.app.ActionBarActivity;
import android.view.View;
import android.widget.Button;
import android.widget.TextView;
import android.widget.Toast;

import java.io.IOException;
import java.io.InputStream;
import java.io.OutputStream;
import java.util.UUID;



public class ledControl extends ActionBarActivity {

    Button btnOn, btnOff, btnRPM, btnREF, btnVEL, btnAVE, btnBOR;
    TextView txtArduino,txtString, txtStringLength, sensorView0,
```

```
sensorView1, sensorView2, sensorView3, sensorView4, sensorView5; //,
sensorView6
    Handler bluetoothIn;

    final int handlerState = 0;                        //used to
identify handler message
    private BluetoothAdapter btAdapter = null;
    private BluetoothSocket btSocket = null;
    private StringBuilder recDataString = new StringBuilder();

    private ConnectedThread mConnectedThread;

    // SPP UUID service - this should work for most devices
    private static final UUID BTMODULEUUID =
UUID.fromString("00001101-0000-1000-8000-00805F9B34FB");

    // String for MAC address
    private static String address;

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        setContentView(R.layout.activity_led_control1);

        //Link the buttons and textViews to respective views
        btnOn = (Button) findViewById(R.id.button2);
        btnOff = (Button) findViewById(R.id.button3);
        btnRPM = (Button) findViewById(R.id.button6);
        btnREF = (Button) findViewById(R.id.button5);
        btnVEL = (Button) findViewById(R.id.button7);
        btnAVE = (Button) findViewById(R.id.button8);
        btnBOR = (Button) findViewById(R.id.button9);
        txtString = (TextView) findViewById(R.id.txtString);
        txtStringLength = (TextView) findViewById(R.id.testView1);
        sensorView0 = (TextView) findViewById(R.id.sensorView0);
        sensorView1 = (TextView) findViewById(R.id.sensorView1);
        sensorView2 = (TextView) findViewById(R.id.sensorView2);
        sensorView3 = (TextView) findViewById(R.id.sensorView3);
        sensorView4 = (TextView) findViewById(R.id.sensorView4);
        sensorView5 = (TextView) findViewById(R.id.sensorView5);


        bluetoothIn = new Handler() {
            public void handleMessage(android.os.Message msg) {
                if (msg.what == handlerState) {
//if message is what we want
                    String readMessage = (String) msg.obj;
// msg.arg1 = bytes from connect thread
                    recDataString.append(readMessage);
//keep appending to string until ~
                    int endOfLineIndex = recDataString.indexOf("~");
// determine the end-of-line
                    if (endOfLineIndex > 0) {
// make sure there data before ~
                        String dataInPrint =
recDataString.substring(0, endOfLineIndex);    // extract string
                        txtString.setText("Data Received = " +
dataInPrint);
                        int dataLength = dataInPrint.length();
//get length of data received
```

```
                        txtStringLength.setText("String Length = " +
String.valueOf(dataLength));

                        if (recDataString.charAt(0) == '#')
//if it starts with # we know it is what we are looking for
                        {
                            String sensor0 =
recDataString.substring(1, 7);              //get sensor value from
string between indices 1-5

                            sensorView1.setText(" Reset = " + sensor0
+ ";");  //update the textviews with sensor values
                        }

                        else if (recDataString.charAt(0) == ']')
                        {
                            String sensor1 =
recDataString.substring(1, 4);             //same again...

                            sensorView2.setText(" Bateria = " +
sensor1 + "V");
                        }

                        else if (recDataString.charAt(0) == '?')
                        {
                            String sensor2 =
recDataString.substring(1, 5);

                            sensorView3.setText(" Velocidad = " +
sensor2 + "RPM");
                        }

                        else if (recDataString.charAt(0) == '&')
                        {
                            String sensor3 =
recDataString.substring(1, 4);

                            sensorView0.setText(" Temperatura = " +
sensor3 + "°C");
                        }

                        else if (recDataString.charAt(0) == '%')
                        {
                            String sensor4 =
recDataString.substring(1, 4);

                            sensorView4.setText(" Velocidad = " +
sensor4 + "km/h");
                        }

                        else if (recDataString.charAt(0) == '$')
                        {
                            String sensor5 =
recDataString.substring(1, 2);
                            //String sensor6 =
recDataString.substring(5, 14);

                            sensorView5.setText(" N°Averias = " +
sensor5);
                            //sensorView6.setText(sensor6 + ";");
                        }
```

```
                        recDataString.delete(0,
recDataString.length());               //clear all string data
                        // strIncom =" ";
                        dataInPrint = " ";
                    }
                }
            }
        };

        btAdapter = BluetoothAdapter.getDefaultAdapter();       // get
Bluetooth adapter
        checkBTState();


        // Set up onClick listeners for buttons to send 1 or 0 to turn
on
        btnOff.setOnClickListener(new View.OnClickListener() {
            public void onClick(View v) {
                mConnectedThread.write("BAT<*>");    // Send "0" via
Bluetooth
                Toast.makeText(getBaseContext(), "Turn on BAT",
Toast.LENGTH_SHORT).show();
            }
        });

        btnOn.setOnClickListener(new View.OnClickListener() {
            public void onClick(View v) {
                mConnectedThread.write("RST<*>");    // Send "1" via
Bluetooth
                Toast.makeText(getBaseContext(), "Turn on RST",
Toast.LENGTH_SHORT).show();
            }
        });

        btnRPM.setOnClickListener(new View.OnClickListener() {
            public void onClick(View v) {
                mConnectedThread.write("RPM<*>");    // Send "0" via
Bluetooth
                Toast.makeText(getBaseContext(), "Turn on RPM",
Toast.LENGTH_SHORT).show();
            }
        });

        btnREF.setOnClickListener(new View.OnClickListener() {
            public void onClick(View v) {
                mConnectedThread.write("REF<*>");    // Send "1" via
Bluetooth
                Toast.makeText(getBaseContext(), "Turn on REF",
Toast.LENGTH_SHORT).show();
            }
        });

        btnVEL.setOnClickListener(new View.OnClickListener() {
            public void onClick(View v) {
                mConnectedThread.write("VEL<*>");    // Send "0" via
Bluetooth
                Toast.makeText(getBaseContext(), "Turn on VEL",
Toast.LENGTH_SHORT).show();
            }
        });
```

```
        btnAVE.setOnClickListener(new View.OnClickListener() {
            public void onClick(View v) {
                mConnectedThread.write("AVE<*>");    // Send "1" via
Bluetooth
                Toast.makeText(getBaseContext(), "Turn on AVE",
Toast.LENGTH_SHORT).show();
            }
        });

        btnBOR.setOnClickListener(new View.OnClickListener() {
            public void onClick(View v) {
                mConnectedThread.write("BOR<*>");    // Send "0" via
Bluetooth
                Toast.makeText(getBaseContext(), "Turn on BOR",
Toast.LENGTH_SHORT).show();
            }
        });
    }


    private BluetoothSocket createBluetoothSocket(BluetoothDevice
device) throws IOException {

        return
device.createRfcommSocketToServiceRecord(BTMODULEUUID);
        //creates secure outgoing connecetion with BT device using
UUID
    }

    @Override
    public void onResume() {
        super.onResume();

        //Get MAC address from DeviceListActivity via intent
        Intent intent = getIntent();

        //Get the MAC address from the DeviceListActivty via EXTRA
        address = intent.getStringExtra(DeviceList.EXTRA_ADDRESS);

        //create device and set the MAC address
        BluetoothDevice device = btAdapter.getRemoteDevice(address);

        try {
            btSocket = createBluetoothSocket(device);
        } catch (IOException e) {
            Toast.makeText(getBaseContext(), "Socket creation failed",
Toast.LENGTH_LONG).show();
        }
        // Establish the Bluetooth socket connection.
        try
        {
            btSocket.connect();
        } catch (IOException e) {
            try
            {
                btSocket.close();
            } catch (IOException e2)
            {
                //insert code to deal with this
            }
```

```
        }
        mConnectedThread = new ConnectedThread(btSocket);
        mConnectedThread.start();

        //I send a character when resuming.beginning transmission to
check device is connected
        //If it is not an exception will be thrown in the write method
and finish() will be called
        mConnectedThread.write("x");
    }

    @Override
    public void onPause()
    {
        super.onPause();
        try
        {
            //Don't leave Bluetooth sockets open when leaving activity
            btSocket.close();
        } catch (IOException e2) {
            //insert code to deal with this
        }
    }

    //Checks that the Android device Bluetooth is available and
prompts to be turned on if off
    private void checkBTState() {

        if(btAdapter==null) {
            Toast.makeText(getBaseContext(), "Device does not support
bluetooth", Toast.LENGTH_LONG).show();
        } else {
            if (btAdapter.isEnabled()) {
            } else {
                Intent enableBtIntent = new
Intent(BluetoothAdapter.ACTION_REQUEST_ENABLE);
                startActivityForResult(enableBtIntent, 1);
            }
        }
    }

    //create new class for connect thread
    private class ConnectedThread extends Thread {
        private final InputStream mmInStream;
        private final OutputStream mmOutStream;

        //creation of the connect thread
        public ConnectedThread(BluetoothSocket socket) {
            InputStream tmpIn = null;
            OutputStream tmpOut = null;

            try {
                //Create I/O streams for connection
                tmpIn = socket.getInputStream();
                tmpOut = socket.getOutputStream();
            } catch (IOException e) { }

            mmInStream = tmpIn;
            mmOutStream = tmpOut;
        }
```

```
        public void run() {
            byte[] buffer = new byte[256];
            int bytes;

            // Keep looping to listen for received messages
            while (true) {
                try {
                    bytes = mmInStream.read(buffer);            //read
bytes from input buffer
                    String readMessage = new String(buffer, 0, bytes);
                    // Send the obtained bytes to the UI Activity via
handler
                    bluetoothIn.obtainMessage(handlerState, bytes, -1,
readMessage).sendToTarget();
                } catch (IOException e) {
                    break;
                }
            }
        }
        //write method
        public void write(String input) {
            byte[] msgBuffer = input.getBytes();           //converts
entered String into bytes
            try {
                mmOutStream.write(msgBuffer);                //write
bytes over BT connection via outstream
            } catch (IOException e) {
                //if you cannot write, close the application
                Toast.makeText(getBaseContext(), "Connection Failure",
Toast.LENGTH_LONG).show();
                finish();

            }
        }
    }
}
```

Listado III.1. ***Código Java Activity Principal***

Anexo IV

Android Studio, Código Layout Activity Principal

A continuación se muestra (ver Listado IV.1) el código Layout integro de Android Studio para la Activity Principal, de la aplicación desarrollada:

```
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
android:layout_width="match_parent"
    android:layout_height="match_parent"
android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    android:paddingBottom="@dimen/activity_vertical_margin"
tools:context="com.led.led.ledControl"
    android:background="@drawable/contador1"
    android:orientation="vertical"
    android:weightSum="10">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Diagnosis Control"
        android:id="@+id/textView2"
        android:layout_alignParentTop="true"
        android:layout_alignParentLeft="true"
        android:layout_alignParentStart="true"
        android:textColor="#750A0A"/>

    <LinearLayout
```

```
xmlns:android="http://schemas.android.com/apk/res/android"
        android:orientation="horizontal"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:layout_below="@+id/textView2">

      <Button
         android:layout_width="wrap_content"
         android:layout_height="wrap_content"
         android:text="1.RST"
         android:id="@+id/button2"
         android:layout_marginTop="10dp"
         android:layout_alignParentLeft="true"
         android:layout_alignParentStart="true"/>

      <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="2.BAT"
        android:id="@+id/button3"
        android:layout_marginTop="10dp"
        android:layout_alignParentLeft="true"
        android:layout_alignParentStart="true"/>

      <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="3.RPM"
        android:id="@+id/button6"
        android:layout_marginTop="10dp"
        android:layout_below="@+id/button3"
        android:layout_alignParentLeft="true"
        android:layout_alignParentStart="true"/>

    </LinearLayout>

    <LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
        android:orientation="horizontal"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content">

      <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="4.REF"
        android:id="@+id/button5"
        android:layout_marginTop="10dp"
        android:layout_alignParentLeft="true"
        android:layout_alignParentStart="true"/>

      <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="5.VEL"
        android:id="@+id/button7"
        android:layout_marginTop="10dp"
        android:layout_alignParentLeft="true"
        android:layout_alignParentStart="true"/>

      <Button
```

```
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="6.AVE"
        android:id="@+id/button8"
        android:layout_marginTop="10dp"
        android:layout_alignParentLeft="true"
        android:layout_alignParentStart="true"/>


    </LinearLayout>

      <TextView
        android:id="@+id/sensorView1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_below="@+id/sensorView0"
        android:layout_centerHorizontal="true"
        android:text="Reset=...."
        android:textAppearance="?android:attr/textAppearanceMedium"
        android:background="#FFFFFF"/>

      <TextView
        android:id="@+id/sensorView2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_below="@+id/sensorView1"
        android:layout_centerHorizontal="true"
        android:text="Bateria = ...."
        android:textAppearance="?android:attr/textAppearanceMedium"
        android:background="#FFFFFF"/>

      <TextView
        android:id="@+id/sensorView3"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_below="@+id/sensorView2"
        android:layout_centerHorizontal="true"
        android:text="Velocidad = ...."
        android:textAppearance="?android:attr/textAppearanceMedium"
        android:background="#FFFFFF"/>

      <TextView
        android:id="@+id/sensorView0"
        android:layout_width="wrap_content"
        android:layout_centerHorizontal="true"
        android:layout_height="wrap_content"
        android:layout_below="@+id/sensorView3"
        android:text="Temperatura = ...."
        android:textAppearance="?android:attr/textAppearanceMedium"
        android:background="#FFFFFF"/>

     <TextView
        android:id="@+id/sensorView4"
        android:layout_width="wrap_content"
        android:layout_centerHorizontal="true"
        android:layout_height="wrap_content"
        android:layout_below="@+id/sensorView0"
        android:text="Velocidad = ...."
        android:textAppearance="?android:attr/textAppearanceMedium"
        android:background="#FFFFFF"/>
```

```
        <TextView
          android:id="@+id/sensorView5"
          android:layout_width="wrap_content"
          android:layout_centerHorizontal="true"
          android:layout_height="wrap_content"
          android:layout_below="@+id/sensorView4"
          android:text="N°Averias = ...."
          android:textAppearance="?android:attr/textAppearanceMedium"
          android:background="#FFFFFF"/>

        <TextView
          android:id="@+id/testView1"
          android:layout_width="wrap_content"
          android:layout_height="wrap_content"
          android:layout_above="@+id/txtString"
          android:text=""
          android:textColor="#002BFF"
          android:textSize="15sp" />

        <TextView
          android:id="@+id/txtString"
          android:layout_width="wrap_content"
          android:layout_height="wrap_content"
          android:layout_alignLeft="@+id/testView1"
          android:layout_alignParentBottom="true"
          android:text=""
          android:textColor="#002BFF"
          android:textSize="15sp" />

       <Button
          android:layout_width="wrap_content"
          android:layout_height="wrap_content"
          android:text="BOR"
          android:id="@+id/button9"
          android:layout_marginTop="10dp"
          android:layout_alignParentLeft="true"
          android:layout_alignParentStart="true"/>

</LinearLayout>
```

Listado IV.1. ***Código Layout Activity Principal***

Anexo V

Android Studio, Código Java Activity Bluetooth

A continuación se muestra (ver Listado V.1) el código Java integro de Android Studio para la Activity Bluetooth, de la aplicación desarrollada:

```
package com.led.led;

import android.content.Intent;
import android.support.v7.app.ActionBarActivity;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.widget.AdapterView;
import android.widget.ArrayAdapter;
import android.widget.Button;
import android.widget.ListView;
import android.bluetooth.BluetoothAdapter;
import android.bluetooth.BluetoothDevice;
import android.widget.TextView;
import android.widget.Toast;

import java.util.ArrayList;
import java.util.Set;


public class DeviceList extends ActionBarActivity
{
    //Declaramos Los Componentes
    Button btnVinculados;
    ListView listaDispositivos;
```

```
    //Bluetooth
    private BluetoothAdapter myBluetooth = null;
    private Set<BluetoothDevice> dispVinculados;
    public static String EXTRA_ADDRESS = "device_address";

    @Override
    protected void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_device_list);

        //Declaramos nuestros componenetes ralcionandolos con los del
layout
        btnVinculados = (Button)findViewById(R.id.button);
        listaDispositivos = (ListView)findViewById(R.id.listView);

        //Comprobamos que el dispositivo tiene bluetooth
        myBluetooth = BluetoothAdapter.getDefaultAdapter();

        if(myBluetooth == null)
        {
            //Mostramos un mensaje, indicando al usuario que no tiene
conexión bluetooth disponible
            Toast.makeText(getApplicationContext(), "Bluetooth no
disponible", Toast.LENGTH_LONG).show();

            //finalizamos la aplicación
            finish();
        }
        else if(!myBluetooth.isEnabled())
        {
                //Preguntamos al usuario si desea encender el
bluetooth
                Intent turnBTon = new
Intent(BluetoothAdapter.ACTION_REQUEST_ENABLE);
                startActivityForResult(turnBTon,1);
        }

        btnVinculados.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v)
            {
                listaDispositivosvinculados();
            }
        });

    }

    private void listaDispositivosvinculados()
    {
        dispVinculados = myBluetooth.getBondedDevices();
        ArrayList list = new ArrayList();

        if (dispVinculados.size()>0)
        {
            for(BluetoothDevice bt : dispVinculados)
            {
                list.add(bt.getName() + "\n" + bt.getAddress());
//Obtenemos los nombres y direcciones MAC de los disp. vinculados
            }
        }
```

```
        else
        {
            Toast.makeText(getApplicationContext(), "No se han
encontrado dispositivos vinculados", Toast.LENGTH_LONG).show();
        }

        final ArrayAdapter adapter = new
ArrayAdapter(this,android.R.layout.simple_list_item_1, list);
        listaDispositivos.setAdapter(adapter);
        listaDispositivos.setOnItemClickListener(myListClickListener);

    }

    private AdapterView.OnItemClickListener myListClickListener = new
AdapterView.OnItemClickListener()
    {
        public void onItemClick (AdapterView<?> av, View v, int arg2,
long arg3)
        {
            // Get the device MAC address, the last 17 chars in the
View
            String info = ((TextView) v).getText().toString();
            String address = info.substring(info.length() - 17);

            // Make an intent to start next activity.
            Intent i = new Intent(DeviceList.this, ledControl.class);

            //Change the activity.
            i.putExtra(EXTRA_ADDRESS, address); //this will be
received at ledControl (class) Activity
            startActivity(i);
        }
    };


    @Override
    public boolean onCreateOptionsMenu(Menu menu)
    {
        getMenuInflater().inflate(R.menu.menu_device_list, menu);
        return true;
    }

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        // Handle action bar item clicks here. The action bar will
        // automatically handle clicks on the Home/Up button, so long
        // as you specify a parent activity in AndroidManifest.xml.
        int id = item.getItemId();
        //noinspection SimplifiableIfStatement
        if (id == R.id.action_settings) {
            return true;
        }
        return super.onOptionsItemSelected(item);
    }
}
```

Listado V.1. ***Código Java Activity Bluetooth***

Anexo VI

Android Studio, Código Layout Activity Bluetooth

A continuación se muestra (ver Listado VI.1) el código Layout integro de Android Studio para la Activity Bluetooth, de la aplicación desarrollada:

```
<RelativeLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
android:layout_width="match_parent"
    android:layout_height="match_parent"
android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    android:background="@drawable/bluetooth1"
    android:paddingBottom="@dimen/activity_vertical_margin"
tools:context=".DeviceList">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Dispositivos Vinculados"
        android:id="@+id/textView"
        android:layout_alignParentTop="true"
        android:layout_alignParentLeft="true"
        android:layout_alignParentStart="true" />

    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Dispositivos Vinculados"
```

```
        android:id="@+id/button"
        android:layout_alignParentBottom="true"
        android:layout_alignParentLeft="true"
        android:layout_alignParentStart="true"
        android:layout_alignParentRight="true"
        android:layout_alignParentEnd="true" />

    <ListView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/listView"
        android:layout_centerHorizontal="true"
        android:layout_above="@+id/button"
        android:layout_below="@+id/textView" />
</RelativeLayout>
```

Listado VI.1. ***Código Layout Activity Bluetooth***

Anexo VII

Códigos de Averías

A continuación se muestra una lista de códigos de fallos para OBD-II:

- P0001 Fallo eléctrico en el regulador de combustible Circuito Abierto
- P0002 Fallo eléctrico en el regulador de combustible Circuito pobre en rendimiento.
- P0003 Fallo eléctrico en el regulador de combustible Circuito Pobre
- P0004 Fallo eléctrico en el regulador de combustible Circuito Alta entrada
- P0005 Válvula de corte de combustible circuito abierto.
- P0006 Válvula de corte de combustible circuito bajo.
- P0007 Válvula de corte de combustible circuito alto.
- P0008 Error de posición entre cigüeñal y árbol de levas (Arbol1)
- P0009 Error de posición entre cigüeñal y árbol de levas (Arbol2)
- P0010 Fallo en la válvula de la distribución variable 1 circuito abierto
- P0011 Árbol de levas 1 Sincronización adelantada.
- P0012 Árbol de levas 1 Sincronización retrasada.
- P0013 Posición del árbol de levas 2 Actuador falla
- P0014 Árbol de levas 2 Sincronización adelantada.
- P0015 Árbol de levas 2 Sincronización retrasada
- P0016 Posición del Cigüeñal Relación incorrecta con el árbol de levas 1
- P0017 Posición del Cigüeñal Relación incorrecta con el árbol de levas 2
- P0018 Posición del Cigüeñal Relación incorrecta con el árbol de levas 3
- P0019 Posición del Cigüeñal Relación incorrecta con el árbol de levas 4
- P0020 Fallo en la válvula de la distribución variable 2 Circuito abierto.
- P0021 Fallo en la válvula de la distribución variable 1. Posición avanzada

- P0022 Fallo en la válvula de la distribución variable 1. Posición retrasada.
- P0023 Fallo en la válvula de la distribución variable 2. Circuito abierto.
- P0024 Fallo en la válvula de la distribución variable 2. Posición avanzada
- P0025 Fallo en la válvula de la distribución variable 2. Posición retrasada.
- P0026 Árbol de levas, solenoide de admisión no funciona cuando se ordena. Fuera de rango.
- P0027 Árbol de levas, solenoide de escape no funciona cuando se ordena. Rendimiento
- P0028 Árbol de levas 1 solenoide de admisión no funciona cuando se ordena. Fuera de rango.
- P0029 Árbol de levas 1, solenoide de escape no funciona cuando se ordena. Rendimiento
- P0030 Fallo en el calentador de la sonda lambda Banco 1 sensor 1
- P0031 Fallo en el calentador de la sonda lambda. Tensión baja. Banco 1 sensor 1
- P0032 Fallo en el calentador de la sonda lambda. Tensión alta. Banco 1 sensor 1
- P0033 Fallo de la válvula de derivación del turbocompresor. Válvula de descarga
- P0034 Fallo de la válvula de derivación del turbo. Válvula de descarga, circuito bajo
- P0035 Fallo de la válvula de derivación del turbo. Válvula de descarga, circuito Alto
- P0036 Fallo sonda lambda circuito del calentador Banco 1 sensor 2
- P0037 Fallo calentador sonda lambda circuito bajo. Banco 1 sensor 2
- P0038 Fallo calentador sonda lambda circuito alto. Banco 1 sensor 2
- P0039 Válvula derivación del turbo fuera de rango
- P0040 Señales de la sonda lambda cambiadas Banco 1 Sensor 1 / Banco 2 Sensor 1
- P0041 Señales de la sonda lambda cambiadas 1 Sensor 2 / Banco 2 Sensor 2
- P0042 Circuito defectuoso en el calentador de la sonda lambda Sensor 3 bloque 1
- P0043 Circuito bajo en el calentador de la sonda lambda Sensor 3 bloque 1
- P0044 Circuito alto en el calentador de la sonda lambda Sensor 3 bloque 1
- P0045 Solenoide del turbocompresor circuito abierto
- P0046 Solenoide del turbocompresor circuito Rendimiento incorrecto
- P0047 Solenoide del turbocompresor circuito bajo
- P0048 Solenoide del turbocompresor circuito alta
- P0049 Exceso de velocidad en el turbo
- P0050 Calentador sonda lambda (Sensor 1 bloque 2) circuito defectuoso
- P0051 Calentador sonda lambda (Sensor 1 bloque 2) circuito bajo
- P0052 Calentador sonda lambda (Sensor 1 bloque 2) circuito alto
- P0053 Calentador sonda lambda (Sensor 1 bloque 1) – resistencia
- P0054 Calentador sonda lambda (Sensor 2 bloque 1) – resistencia
- P0055 Calentador sonda lambda (Sensor 3 bloque 1) – resistencia
- P0056 Calentador sonda lambda (Sensor 2 bloque 2) circuito defectuoso
- P0057 Calentador sonda lambda (Sensor 2 bloque 2) señal baja
- P0058 Calentador sonda lambda (Sensor 2 bloque 2) señal alta
- P0059 Calentador sonda lambda (Sensor 1 bloque 2) resistencia.
- P0060 Calentador sonda lambda (Sensor 2 bloque 2) resistencia.

- P0061 Calentador sonda lambda (Sensor 3 bloque 2) resistencia.
- P0062 Calentador sonda lambda (Sensor 3 bloque 2) circuito defectuoso
- P0063 Calentador sonda lambda (Sensor 3 bloque 2) señal baja
- P0064 Calentador sonda lambda (Sensor 3 bloque 2) señal alta
- P0065 Rendimiento incorrecto del inyector de aire asistido.
- P0066 inyector de aire asistido , señal baja
- P0067 inyector de aire asistido , señal alta
- P0068 MAP / MAF y mariposa no coinciden en los datos
- P0069 No coinciden los datos del sensor de presión barométrica con presión absoluta.
- P0070 Fallo en el circuito del sensor de temperatura.
- P0071 Rendimiento incorrecto del sensor de temperatura
- P0072 Señal baja sensor de temperatura 1
- P0073 Señal baja sensor de temperatura 2
- P0074 Interrupción en la señal del sensor de temperatura
- P0075 Solenoide (BLOQUE 1) válvula de admisión –circuito mal
- P0076 Solenoide (BLOQUE 1) válvula de admisión –señal baja
- P0077 Solenoide (BLOQUE 1) válvula de admisión –señal alta
- P0078 Solenoide (BLOQUE 1) válvula de escape –circuito mal
- P0079 Solenoide (BLOQUE 1) válvula de escape–señal baja
- P0080 Solenoide (BLOQUE 1) válvula de escape–señal alta
- P0081 Solenoide (BLOQUE 2) válvula de admisión –circuito mal
- P0082 Solenoide (BLOQUE 2) válvula de admisión –señal baja
- P0083 Solenoide (BLOQUE 2) válvula de admisión –señal alta
- P0084 Solenoide (BLOQUE 2) válvula de escape –circuito mal
- P0085 Solenoide (BLOQUE 2) válvula de escape–señal alta
- P0086 Solenoide (BLOQUE 2) válvula de escape–señal baja
- P0087 Presión de combustible demasiado baja.
- P0088 Presión de combustible demasiado alta
- P0089 Rendimiento incorrecto en el regulador de presión de combustible 1.
- P0090 Regulador de presión de combustible 1 Circuito abierto
- P0091 Regulador de presión de combustible 1 Cortocircuito a masa
- P0092 Regulador de presión de combustible 1 Cortocircuito a positivo
- P0093 Fuga grande en el sistema de combustible
- P0094 Fuga pequeña en el sistema de combustible
- P0095 Sensor de temperatura 2 de la admisión. Circuito defectuoso
- P0096 Sensor de temperatura 2 de la admisión. Rendimiento incorrecto
- P0097 Sensor de temperatura 2 de la admisión. Señal baja
- P0098 Sensor de temperatura 2 de la admisión. Señal alta
- P0099 Sensor de temperatura 2 de la admisión. Circuito intermitente o errático
- P0100 Fallo caudalimetro Circuito de aire incorrecto.
- P0101 Rango y valores incorrectos caudalimetro.
- P0102 Caudalimetro, Señal baja

- P0103 Caudalimetro, Señal alta
- P0104 Caudalimetro error intermitente.
- P0105 Sensor de presión absoluta/presión barométrica Circuito defectuoso
- P0106 Rendimiento incorrecto sensor de presión absoluta / sensor de presión barométrica.
- P0107 Señal baja sensor de presión absoluta/ sensor de presión barométrica
- P0108 Señal alta sensor de presión absoluta/sensor de presión barométrica
- P0109 Señal intermitente sensor de presión absoluta/sensor de presión barométrica
- P0110 Circuito defectuoso sensor 1 temperatura aire admisión.
- P0111 Rendimiento incorrecto sensor temperatura 1 aire admisión.
- P0112 Señal baja sensor temperatura 1 aire admisión.
- P0113 Señal alta sensor temperatura 1 aire admisión.
- P0114 Señal intermitente sensor temperatura 1aire admisión.
- P0115 Circuito defectuoso sensor de temperatura del anticongelante
- P0116 Rendimiento incorrecto Circuito defectuoso sensor de temperatura del anticongelante
- P0117 Señal baja sensor de temperatura del anticongelante
- P0118 Señal alta sensor de temperatura del anticongelante
- P0119 Señal intermitente sensor de temperatura del anticongelante
- P0120 Circuito defectuoso sensor posición pedal acelerador 1/mariposa 1
- P0121 Rendimiento incorrecto del sensor posición pedal acelerador 1/mariposa 1
- P0122 Señal baja del sensor posición pedal acelerador 1/mariposa 1
- P0123 Señal alta del sensor posición pedal acelerador 1/mariposa 1
- P0124 Señal intermitente del sensor posición pedal acelerador 1/mariposa 1
- P0125 Temperatura del refrigerante insuficiente para bucle cerrado de combustible
- P0126 Temperatura del refrigerante insuficiente para un funcionamiento estable
- P0127 Temperatura del aire de admisión demasiado alta
- P0128 Circuito defectuoso sensor de anticongelante en el termostato
- P0129 Presión barométrica demasiado baja
- P0130 Sonda lambda (Sensor 1 bloque 1) Circuito defectuoso
- P0131 Sonda lambda (Sensor 1 bloque 1) Baja tensión
- P0132 Sonda lambda (Sensor 1 bloque 1) Alta tensión
- P0133 Sonda lambda (Sensor 1 bloque 1) Respuesta lenta
- P0134 Sonda lambda (Sensor 1 bloque 1) no se detectó actividad
- P0135 Sonda lambda (Sensor 1 bloque 1) rendimiento incorrecto
- P0136 Sonda lambda (Sensor 2 bloque 1) circuito defectuoso
- P0137 Sonda lambda (Sensor 2 bloque 1) Baja tensión
- P0138 Sonda lambda (Sensor 2 bloque 1) Alta tensión
- P0139 Sonda lambda (Sensor 2 bloque 1) Respuesta lenta
- P0140 Sonda lambda (Sensor 2 bloque 1) no se detectó actividad
- P0141 Circuito defectuoso Sonda lambda (Sensor 2 bloque 1)
- P0142 Sonda lambda (Sensor 3 bloque 1) – circuito defectuoso
- P0143 Sonda lambda (Sensor 3 bloque 1) Baja tensión

- P0144 Sonda lambda (Sensor 3 bloque 1) Alta tensión
- P0145 Sonda lambda (Sensor 3 bloque 1) Respuesta lenta
- P0146 Sonda lambda (Sensor 3 bloque 1) no se detectó actividad
- P0147 Fallo en el rendimiento Sonda lambda (Sensor 3 bloque 1)
- P0148 Error en la alimentación de combustible
- P0149 Error del reglaje de combustible
- P0150 Sonda lambda (Sensor 1 bloque 2) – circuito defectuoso
- P0151 Sonda lambda (Sensor 1 bloque 2) baja tensión
- P0152 Sonda lambda (Sensor 1 bloque 2) Alto voltaje
- P0153 Sonda lambda (Sensor 1 bloque 2) Respuesta lenta
- P0154 Sonda lambda (Sensor 1 bloque 2) no se detectó actividad
- P0155 Circuito defectuoso Sonda lambda (Sensor 1 bloque 2)
- P0156 Sonda lambda (Sensor 2 bloque 2) – circuito defectuoso
- P0157 Sonda lambda (Sensor 2 bloque 2) baja tensión
- P0158 Sonda lambda (Sensor 2 bloque 2) Alto voltaje
- P0159 Sonda lambda (Sensor 2 bloque 2) Respuesta lenta
- P0160 Sonda lambda (Sensor 2 bloque 2) no se detectó actividad
- P0161 Circuito defectuoso Sonda lambda (Sensor 2 bloque 2)
- P0162 Sonda lambda (Sensor 3 bloque 2) circuito defectuoso
- P0163 Sonda lambda (Sensor 3 bloque 2) baja tensión
- P0164 Sonda lambda (Sensor 3 bloque 2) Alto voltaje
- P0165 Sonda lambda (Sensor 3 bloque 2) Respuesta lenta
- P0166 Sonda lambda (Sensor 3 bloque 2) no se detectó actividad
- P0167 Circuito defectuoso Sonda lambda (Sensor 3 bloque 2)
- P0168 Temperatura combustible demasiado alta
- P0169 Composición del combustible incorrecto.
- P0170 Regulación inyección en el bloque 1 (circuito defectuoso)
- P0171 Regulación inyección en el bloque 1 (inyección pobre)
- P0172 Regulación inyección en el bloque 1 (inyección rica)
- P0173 Regulación inyección en el bloque 2 (circuito defectuoso)
- P0174 Regulación inyección en el bloque 2 (inyección pobre)
- P0175 Regulación inyección en el bloque 2 (inyección rica)
- P0176 Fallo en el circuito del sensor de composición de combustible
- P0177 Rango incorrecto del sensor de composición de combustible
- P0178 Señal baja del sensor de composición de combustible
- P0179 Señal alta del sensor de composición de combustible
- P0180 Circuito defectuoso del sensor de composición de combustible
- P0181 Sensor de temperatura de combustible 1 (Circuito defectuoso)
- P0182 Señal baja sensor 1 de temperatura de combustible.
- P0183 Señal alta sensor 1 de temperatura de combustible.
- P0184 Señal intermitente sensor 1 de temperatura de combustible.
- P0185 Circuito defectuoso sensor 2 de temperatura de combustible
- P0186 Valores fuera de rango en el sensor 2 de temperatura de combustible

- P0187 Señal baja en el sensor 2 de temperatura de combustible
- P0188 Señal alta en el sensor 2 de temperatura de combustible
- P0189 Fallo intermitente en el sensor 2 de temperatura de combustible
- P0190 Circuito defectuoso en el sensor de presión de la rampa de combustible
- P0191 Rendimiento incorrecto en el sensor de presión de la rampa de combustible
- P0192 Señal baja en el sensor de presión de la rampa de combustible
- P0193 Señal alta en el sensor de presión de la rampa de combustible
- P0194 Señal intermitente en el sensor de presión de la rampa de combustible
- P0195 Sensor de temperatura del aceite Circuito defectuoso.
- P0196 Rendimiento y rango incorrecto sensor de temperatura del aceite
- P0197 Señal baja Sensor de temperatura del aceite
- P0198 Señal alta Sensor de temperatura del aceite
- P0199 Señal intermitente Sensor de temperatura del aceite
- P0200 Inyectores (Circuito defectuoso)
- P0201 Inyector del cilindro 1 –Circuito defectuoso
- P0202 Inyector del cilindro 2 –Circuito defectuoso
- P0203 Inyector del cilindro 3 –Circuito defectuoso
- P0204 Inyector del cilindro 4 –Circuito defectuoso
- P0205 Inyector del cilindro 5 –Circuito defectuoso
- P0206 Inyector del cilindro 6 –Circuito defectuoso
- P0207 Inyector del cilindro 7 –Circuito defectuoso
- P0208 Inyector del cilindro 8 –Circuito defectuoso
- P0209 Inyector del cilindro 9 –Circuito defectuoso
- P0210 Inyector del cilindro 10 –Circuito defectuoso
- P0211 Inyector del cilindro 11 –Circuito defectuoso
- P0212 Inyector del cilindro 12 –Circuito defectuoso
- P0213 Inyector de arranque en frió 1 –Circuito defectuoso
- P0214 Inyector de arranque en frió 2- Circuito defectuoso
- P0215 Circuito defectuoso solenoide de corte de combustible
- P0216 Reglaje de la inyección _(Circuito defectuoso)
- P0217 Sobrecalentamiento del motor Sensor de temperatura anticongelante
- P0218 Sobrecalentamiento liquido de la transmisión
- P0219 Sobre régimen del motor.
- P0220 Sensor de posición del acelerador 2/mariposa 2- Circuito defectuoso
- P0221 Rendimiento incorrecto Sensor de posición del acelerador 2/mariposa 2
- P0222 Señal baja Sensor de posición del acelerador 2/mariposa 2
- P0223 Señal alta Sensor de posición del acelerador 2/mariposa 2
- P0224 Señal intermitente Sensor de posición del acelerador 2/mariposa 2
- P0225 Sensor de posición del acelerador 3/mariposa 3- Circuito defectuoso
- P0226 Rendimiento incorrecto Sensor de posición del acelerador 3/mariposa 3
- P0227 Señal baja Sensor de posición del acelerador 3/mariposa 3
- P0228 Señal alta Sensor de posición del acelerador 3/mariposa 3
- P0229 Señal intermitente Sensor de posición del acelerador 3/mariposa 3

- P0230 Relé de la bomba de combustible- Circuito defectuoso
- P0231 Señal baja del relé de la bomba de combustible secundaria
- P0232 Señal alta del relé de la bomba de combustible secundaria
- P0233 Señal intermitente del relé de la bomba de combustible secundaria
- P0234 Sobrealimentación del turbo. Limite excedido
- P0235 Sobrealimentación del turbo Limite previsto no alcanzado
- P0236 Rendimiento incorrecto sensor de presión absoluta colector 1 del turbo
- P0237 Señal baja sensor de presión absoluta colector 1 del turbo
- P0238 Señal alta sensor de presión absoluta colector 1del turbo.
- P0239 Circuito defectuoso sensor de presión absoluta colector 2 del turbo
- P0240 Rendimiento incorrecto sensor de presión absoluta colector 2 del turbo
- P0241 Señal baja sensor de presión absoluta colector 2 del turbo
- P0242 Señal alta sensor de presión absoluta colector 2 del turbo
- P0243 Válvula de descarga turbo 1 (WASTEGATE) circuito defectuoso
- P0244 Rendimiento incorrecto Válvula de descarga turbo 1 (WASTEGATE)
- P0245 Señal baja Válvula de descarga turbo 1 (WASTEGATE)
- P0246 Señal alta Válvula de descarga turbo 1 (WASTEGATE)
- P0247 Válvula de descarga turbo 2 (WASTEGATE) circuito defectuoso
- P0248 Rendimiento incorrecto Válvula de descarga turbo 2 (WASTEGATE)
- P0249 Señal baja Válvula de descarga turbo 2 (WASTEGATE)
- P0250 Señal alta Válvula de descarga turbo 2 (WASTEGATE)
- P0251 Circuito defectuoso de la bomba de inyección 1 (Árbol levas/rotor)
- P0252 Rango/funcionamiento incorrecto de la bomba de inyección 1 (Árbol levas/rotor)
- P0253 Señal baja de la bomba de inyección 1 (Árbol levas/rotor)
- P0254 Señal alta de la bomba de inyección 1 (Árbol levas/rotor)
- P0255 Señal intermitente de la bomba de inyección 1 (Árbol levas/rotor)
- P0256 Circuito defectuoso de la bomba de inyección 2 (Árbol levas/rotor)
- P0257 Rango/funcionamiento incorrecto de la bomba de inyección 2 (Árbol levas/rotor)
- P0258 Señal baja de la bomba de inyección 2 (Árbol levas/rotor)
- P0259 Señal alta de la bomba de inyección 2 (Árbol levas/rotor)
- P0260 Señal intermitente de la bomba de inyección 2 (Árbol levas/rotor)
- P0261 Señal baja inyector cilindro 1
- P0262 Señal alta inyector cilindro 1
- P0263 Fallo en el equilibro correcto de combustible en inyector cilindro 1
- P0264 Señal baja inyector cilindro 2
- P0265 Señal alta inyector cilindro 2
- P0266 Fallo en el equilibro correcto de combustible en inyector cilindro 2
- P0267 Señal baja inyector cilindro 3
- P0268 Señal alta inyector cilindro 3
- P0269 Fallo en el equilibro correcto de combustible en inyector cilindro 3
- P0270 Señal baja inyector cilindro 4
- P0271 Señal alta inyector cilindro 4

- P0272 Fallo en el equilibro correcto de combustible en inyector cilindro 4
- P0273 Señal baja inyector cilindro 5
- P0274 Señal alta inyector cilindro 5
- P0275 Fallo en el equilibro correcto de combustible en inyector cilindro 5
- P0276 Señal baja inyector cilindro 6
- P0277 Señal alta inyector cilindro 6
- P0278 Fallo en el equilibro correcto de combustible en inyector cilindro 6
- P0279 Señal baja inyector cilindro 7
- P0280 Señal alta inyector cilindro 7
- P0281 Fallo en el equilibro correcto de combustible en inyector cilindro 7
- P0282 Señal baja inyector cilindro 8
- P0283 Señal alta inyector cilindro 8
- P0284 Fallo en el equilibro correcto de combustible en inyector cilindro 8
- P0285 Señal baja inyector cilindro 9
- P0286 Señal alta inyector cilindro 9
- P0287 Fallo en el equilibro correcto de combustible en inyector cilindro 9
- P0288 Señal baja inyector cilindro 10
- P0289 Señal alta inyector cilindro 10
- P0290 Fallo en el equilibro correcto de combustible en inyector cilindro 10
- P0291 Señal baja inyector cilindro 11
- P0292 Señal alta inyector cilindro 11
- P0293 Fallo en el equilibro correcto de combustible en inyector cilindro 11
- P0294 Señal baja inyector cilindro 12
- P0295 Señal alta inyector cilindro 12
- P0296 Fallo en el equilibro correcto de combustible en inyector cilindro 12
- P0297 Exceso de velocidad del vehículo. Veras la guardia civil como te pille
- P0298 Temperatura del aceite del motor demasiado alta.
- P0299 Nivel de presión del turbo demasiado bajo
- P0300 Uno o varios cilindros fallan. Falsas explosiones detectadas
- P0301 Cilindro 1 falsa explosión detectada. No funciona correctamente, fallo de encendido
- P0302 Cilindro 2 falsa explosión detectada No funciona correctamente, fallo de encendido
- P0303 Cilindro 3 falsa explosión detectada No funciona correctamente, fallo de encendido
- P0304 Cilindro 4 falsa explosión detectada No funciona correctamente, fallo de encendido
- P0305 Cilindro 5 falsa explosión detectada No funciona correctamente, fallo de encendido
- P0306 Cilindro 6 falsa explosión detectada No funciona correctamente, fallo de encendido
- P0307 Cilindro 7 falsa explosión detectada No funciona correctamente, fallo de encendido

- P0308 Cilindro 8 falsa explosión detectada No funciona correctamente, fallo de encendido
- P0309 Cilindro 9 falsa explosión detectada No funciona correctamente, fallo de encendido
- P0310 Cilindro 10 falsa explosión detectada No funciona correctamente, fallo de encendido
- P0311 Cilindro 11 falsa explosión detectada No funciona correctamente, fallo de encendido
- P0312 Cilindro 12 falsa explosión detectada No funciona correctamente, fallo de encendido
- P0313 Detectado fallo de encendido con bajo nivel de combustible
- P0314 Fallo de encendido en un cilindro (cilindro no especificado)
- P0315 Variación de la posición del cigüeñal con respecto a los valores prefijados de fabrica
- P0316 Fallo de encendido en el arranque (solo hasta 1000 revoluciones)
- P0317 No se encuentra el hardware sensor carretera desnivelada. Usado para posicionar cigüeñal en carreteras
- P0318 Circuito defectuoso sensor de carretera bacheada 1
- P0319 Circuito defectuoso sensor de carretera bacheada 2
- P0320 Circuito defectuoso del sensor de posición del cigüeñal /régimen motor
- P0321 Rendimiento incorrecto del sensor de posición del cigüeñal /régimen motor
- P0322 Sin señal sensor de posición cigüeñal/régimen motor.
- P0323 Circuito intermitente sensor posición del cigüeñal/régimen motor
- P0324 Error en el sistema de detonación. KNOCK sensor
- P0325 Circuito defectuoso sensor detonación 1 (bloque1)
- P0326 Rendimiento incorrecto sensor detonación 1 (bloque1)
- P0327 Señal baja sensor detonación 1 (bloque1)
- P0328 Señal alta sensor detonación 1 (bloque1)
- P0329 Señal intermitente sensor detonación 1 (bloque1)
- P0330 Circuito defectuoso sensor detonación 2 (bloque 2)
- P0331 Rendimiento incorrecto sensor detonación 2 (bloque 2)
- P0332 Señal baja sensor detonación 2 (bloque 2)
- P0333 Señal alta sensor detonación 2 (bloque 2)
- P0334 Señal intermitente sensor detonación 2 (bloque 2)
- P0335 Sensor posición cigüeñal 1 circuito defectuoso
- P0336 Rendimiento incorrecto Sensor posición cigüeñal 1
- P0337 Señal baja Sensor posición cigüeñal 1
- P0338 Señal alta Sensor posición cigüeñal 1
- P0339 Señal intermitente Sensor posición cigüeñal 1
- P0340 Circuito defectuoso sensor posición árbol de levas 1 bloque 1
- P0341 Rango incorrecto sensor posición árbol de levas 1 bloque 1
- P0342 Señal baja sensor posición árbol de levas 1 bloque 1
- P0343 Señal alta sensor posición árbol de levas 1 bloque 1
- P0344 Señal intermitente sensor posición árbol de levas 1 bloque 1

- P0345 Circuito defectuoso sensor posición árbol de levas 1 bloque 2
- P0346 Rendimiento incorrecto sensor posición árbol de levas 1 bloque 2
- P0347 Señal baja sensor posición árbol de levas 1 bloque 2
- P0348 Señal alta sensor posición árbol de levas 1 bloque 2
- P0349 Señal intermitente sensor posición árbol de levas 1 bloque 2
- P0350 Circuito defectuoso Bobina de encendido primaria o secundaria
- P0351 Circuito defectuoso Bobina de encendido A Circuito primario o secundario
- P0352 Circuito defectuoso Bobina de encendido B Circuito primario o secundario
- P0353 Circuito defectuoso Bobina de encendido C Circuito primario o secundario
- P0354 Circuito defectuoso Bobina de encendido D Circuito primario o secundario
- P0355 Circuito defectuoso Bobina de encendido E Circuito primario o secundario
- P0356 Circuito defectuoso Bobina de encendido F Circuito primario o secundario
- P0357 Circuito defectuoso Bobina de encendido G Circuito primario o secundario
- P0358 Circuito defectuoso Bobina de encendido H Circuito primario o secundario
- P0359 Circuito defectuoso Bobina de encendido I Circuito primario o secundario
- P0360 Circuito defectuoso Bobina de encendido J Circuito primario o secundario
- P0361 Circuito defectuoso Bobina de encendido K Circuito primario o secundario
- P0362 Circuito defectuoso Bobina de encendido L Circuito primario o secundario
- P0363 Fallo de encendido. Anulación de alimentación de combustible
- P0364 Reservado
- P0365 Circuito defectuoso en el sensor del árbol de levas 2
- P0366 Rendimiento incorrecto en el sensor del árbol de levas 2
- P0367 Señal baja en el sensor del árbol de levas 2
- P0368 Señal alta en el sensor del árbol de levas 2
- P0369 Señal intermitente en el sensor del árbol de levas 2
- P0370 Señal defectuosa en el reglaje de encendido. 1
- P0371 Demasiados impulsos en la señal del reglaje de encendido 1
- P0372 Muy pocos impulsos en la señal del reglaje del encendido 1
- P0373 Señal intermitente en la señal del reglaje de encendido. 1
- P0374 No hay pulso ni señal en el reglaje del encendido 1
- P0375 Señal defectuosa en el reglaje de encendido. 2
- P0376 Demasiados impulsos en la señal del reglaje de encendido 2
- P0377 Muy pocos impulsos en la señal del reglaje del encendido 2
- P0378 Señal intermitente en la señal del reglaje de encendido. 2
- P0379 No hay pulso ni señal en el reglaje del encendido 2
- P0380 Fallo en los calentadores en el circuito 1
- P0381 Circuito defectuoso en el testigo de los calentadores
- P0382 Fallo en los calentadores en el circuito 2
- P0383 Reservado para SAE J2012
- P0384 Reservado para SAE J2012
- P0385 Sensor de posición de cigüeñal 2 –circuito defectuoso.
- P0386 Rendimiento incorrecto Sensor de posición de cigüeñal 2
- P0387 Señal baja Sensor de posición de cigüeñal 2

- P0388 Señal alta Sensor de posición de cigüeñal 2
- P0389 Señal intermitente Sensor de posición de cigüeñal 2
- P0390 Circuito defectuoso en el sensor del árbol de levas 2
- P0391 Rendimiento incorrecto en el sensor del árbol de levas 2
- P0392 Señal baja en el sensor del árbol de levas 2
- P0393 Señal alta en el sensor del árbol de levas 2
- P0394 Señal intermitente en el sensor del árbol de levas 2
- P0400 Flujo incorrecto en la EGR
- P0401 Poco flujo en la Egr
- P0402 Demasiado flujo en la Egr
- P0403 Circuito defectuoso en la EGR
- P0404 Rendimiento incorrecto de la válvula EGR
- P0405 Señal baja en el sensor 1 de la EGR
- P0406 Señal alta en el sensor 1 de la EGR
- P0407 Señal baja en el sensor 2 de la EGR
- P0408 Señal alta en el sensor 2 de la EGR
- P0409 Circuito defectuoso en el sensor 1 de la EGR
- P0410 Sistema de inyección de aire secundario DEFECTUOSO
- P0411 Flujo incorrecto en el sistema de inyección de aire secundario
- P0412 Circuito defectuoso en la válvula de inyección de aire 1
- P0413 Circuito abierto en la válvula de inyección de aire 1
- P0414 Cortocircuito en la válvula de inyección de aire 1
- P0415 Circuito defectuoso en la válvula de inyección de aire 2
- P0416 Circuito abierto en la válvula de inyección de aire 2
- P0417 Cortocircuito en la válvula de inyección de aire 2
- P0418 Relé defectuoso en la inyección de aire 1
- P0419 Relé defectuoso en la inyección de aire 2
- P0420 Eficiencia del catalizador 1 por debajo del umbral
- P0421 Eficiencia del catalizador delantero 1 por debajo del umbral
- P0422 Eficiencia del catalizador principal por debajo del umbral
- P0423 Eficiencia del catalizador secundario 1 por debajo del umbral
- P0424 Temperatura del catalizador 1 por debajo del umbral
- P0425 El sensor de temperatura del catalizador 1 no está presente o esta defectuoso
- P0426 Rendimiento incorrecto del sensor de temperatura del catalizador 1
- P0427 Señal baja del sensor de temperatura del catalizador 1
- P0428 Señal alta del sensor de temperatura del catalizador 1
- P0429 Circuito defectuoso del sensor de temperatura del catalizador 1
- P0430 Eficiencia por debajo del umbral en el catalizador 2
- P0431 Catalizador delantero 2 eficiencia por debajo del umbral
- P0432 Eficiencia catalizador principal 2 por debajo del umbral
- P0433 Eficiencia temperatura catalizador 2 por debajo del umbral
- P0434 Temperatura por debajo del umbral en el catalizador 2
- P0435 El sensor de temperatura del catalizador 2 no está presente o esta defectuoso

- P0436 Rendimiento incorrecto del sensor de temperatura del catalizador 2
- P0437 Señal baja del sensor de temperatura del catalizador 2
- P0438 Señal alta del sensor de temperatura del catalizador 2
- P0439 Calentador Catalizador Control del Circuito
- P0440 Sistema emisiones evaporadas defectuoso Sistema EVAP
- P0441 Flujo incorrecto en el sistema Evap Sistema emisiones evaporadas
- P0442 Fuga en el sistema EVAP Sistema emisiones evaporadas
- P0443 Circuito defectuoso en la válvula de control de emisiones.
- P0444 Circuito abierto en la válvula de purga del control de emisiones
- P0445 Cortocircuito en la válvula de purga del control de emisiones
- P0446 Circuito defectuoso en la válvula de purga del control de emisiones
- P0447 Circuito abierto en el sistema de control de ventilación de las evaporaciones
- P0448 Cortocircuito en el sistema de control de ventilación de las evaporaciones
- P0449 Circuito defectuoso en el sistema de control de ventilación de las evaporaciones
- P0450 Circuito defectuoso sensor de presión de emisiones por evaporación
- P0451 Rendimiento incorrecto sensor de presión de emisiones por evaporación
- P0452 Señal baja en el sensor de presión de emisiones por evaporación
- P0453 Presión alta en el sensor de presión de emisiones por evaporación
- P0454 Señal intermitente en el sensor de presión de emisiones por evaporación
- P0455 Fuga grande en el sistema emisiones evaporación
- P0456 Fuga pequeña en el sistema emisiones evaporación
- P0457 Fuga detectada en el sistema emisiones evaporación
- P0458 Señal baja en el sistema de control de emisiones por evaporaciones
- P0459 Señal alta en el sistema de control de emisiones por evaporaciones
- P0460 Circuito defectuoso en el sensor de nivel de combustible
- P0461 Rendimiento incorrecto del sensor de nivel de combustible
- P0462 Señal baja Sensor de combustible
- P0463 Señal alta sensor de combustible
- P0464 Señal intermitente en el sensor de combustible.
- P0465 Circuito defectuoso en el sensor de purga del sistema de evaporaciones EVAP
- P0466 Rendimiento incorrecto en el sensor de purga del sistema de evaporaciones EVAP
- P0467 Señal baja en el sensor de purga del sistema de evaporaciones EVAP
- P0468 Señal alta en el sensor de purga del sistema de evaporaciones EVAP
- P0469 Señal intermitente en el sensor de purga del sistema de evaporaciones EVAP
- P0470 Circuito defectuoso en Sensor de presión de gases de escape
- P0471 Rendimiento incorrecto en el sensor de presión de gases de escape
- P0472 Baja presión en el sensor de presión de los gases de escape
- P0473 Alta presión detectada en el sensor de presión de los gases de escape
- P0474 Señal intermitente en el sensor de presión de los gases de escape
- P0475 Circuito defectuoso en el sensor de presión de los gases de escape

- P0476 Rendimiento incorrecto en la válvula reguladora de los gases de escape
- P0477 Señal baja en la válvula reguladora de los gases de escape
- P0478 Señal alta en la válvula reguladora de los gases de escape
- P0479 Señal intermitente en la válvula reguladora de los gases de escape
- P0480 Circuito defectuoso Ventilador refrigerante motor 1
- P0481 Circuito defectuoso ventilador refrigerante motor 2
- P0482 Circuito defectuoso ventilador refrigerante motor 3
- P0483 Ventilador refrigerante motor defectuoso
- P0484 Sobrecarga corriente en ventilador refrigerante motor
- P0485 Ventilador de refrigerante motor deriva a masa- Circuito defectuoso.
- P0486 Sensor válvula egr 2 defectuoso
- P0487 Circuito defectuoso en los gases de recirculación con respecto a la mariposa
- P0488 Rendimiento incorrecto gases de recirculación con respecto a la mariposa.
- P0489 Señal baja en la EGR (recirculación de gases de escape)
- P0490 Señal alta en la EGR (recirculación de gases de escape)
- P0491 Funcionamiento incorrecto en el sistema de inyección de aire secundario en bloque 1
- P0492 Funcionamiento incorrecto en el sistema de inyección de aire secundario en bloque 2
- P0493 Exceso de velocidad del ventilador refrigerante
- P0494 Fallo en la velocidad baja del ventilador de refrigerante
- P0495 Fallo en la velocidad alta del ventilador de refrigerante
- P0496 Flujo purga alto en el sistema de emisiones de evaporación
- P0497 Flujo purga bajo en el sistema de emisiones de evaporación
- P0498 Señal baja en el control de ventilación de las emisiones evaporativas
- P0499 Señal alta en el control de ventilación de las emisiones evaporativas
- P0500 Circuito defectuoso en el sensor de velocidad del vehículo
- P0501 Rendimiento incorrecto en el sensor de velocidad del vehículo
- P0502 Señal baja del sensor de velocidad del vehículo
- P0503 Señal alta/intermitente del sensor de velocidad del vehículo.
- P0504 Relación del interruptor del freno no es correcta
- P0505 Control del ralentí defectuoso
- P0506 Revoluciones inferior a lo previsto en el sistema de control del ralentí
- P0507 Revoluciones superior a lo previsto en el sistema de control del ralentí
- P0508 Señal baja en el control del aire del ralentí.
- P0509 Señal alta en el control del aire del ralentí
- P0510 Circuito defectuoso en el interruptor mariposa cerrada
- P0511 Circuito defectuoso control aire ralentí
- P0512 Circuito petición motor de arranque –Funcionamiento incorrecto
- P0513 Llave incorrecta. Inmovilizador detecta llave incorrecta
- P0514 Rendimiento incorrecto sensor temperatura batería
- P0515 Circuito defectuoso sensor temperatura batería
- P0516 Señal baja sensor temperatura batería
- P0517 Señal alta sensor temperatura batería

- P0518 Señal intermitente control del aire del ralentí
- P0519 Rendimiento incorrecto control del aire del ralentí
- P0520 Circuito defectuoso sensor presión aceite motor
- P0521 Rendimiento incorrecto sensor presión aceite motor
- P0522 Señal baja sensor presión aceite motor
- P0523 Señal alta sensor presión aceite motor
- P0524 Presión de aceite del motor demasiado baja
- P0525 Funcionamiento incorrecto control velocidad crucero
- P0526 Circuito defectuoso sensor velocidad refrigerante motor
- P0527 Rendimiento incorrecto Sensor de velocidad del ventilador del anticongelante
- P0528 No hay señal del ventilador del anticongelante
- P0529 Señal intermitente del sensor de velocidad del ventilador de anticongelante
- P0530 Circuito defectuoso sensor de presión de refrigerante Aire acondicionado
- P0531 Rango incorrecto sensor de presión de refrigerante Aire acondicionado
- P0532 Señal baja sensor de presión de refrigerante Aire acondicionado
- P0533 Señal alta sensor de presión de refrigerante Aire acondicionado
- P0534 Perdida refrigerante aire acondicionado
- P0535 Sensor de temperatura evaporador del aire acondicionado defectuoso
- P0536 Rendimiento incorrecto del sensor de temperatura evaporador del aire acondicionado
- P0537 Señal baja del sensor de temperatura evaporador del aire acondicionado
- P0538 Señal alta del sensor de temperatura evaporador del aire acondicionado
- P0539 Señal intermitente del sensor de temperatura evaporador del aire acondicionado
- P0540 Circuito defectuoso del calentador del aire de la admisión 1
- P0541 Señal baja del calentador del aire de la admisión 1
- P0542 Señal alta del calentador del aire de la admisión 1
- P0543 Circuito abierto del calentador del aire de la admisión 1
- P0544 Sensor temperatura EGR banco 1 defectuoso
- P0545 Señal baja Sensor temperatura EGR banco 1
- P0546 Señal alta Sensor temperatura EGR banco 1
- P0547 Sensor temperatura EGR banco 2 defectuoso
- P0548 Señal baja Sensor temperatura EGR banco 2
- P0549 Señal alta Sensor temperatura EGR banco 2
- P0550 Circuito defectuoso sensor de presión hidráulica en dirección asistida
- P0551 Rango incorrecto sensor de presión hidráulica en dirección asistida
- P0552 Señal baja sensor de presión hidráulica en dirección asistida
- P0553 Señal alta sensor de presión hidráulica en dirección asistida
- P0554 Señal intermitente sensor de presión hidráulica en dirección asistida
- P0555 Circuito defectuoso en el sensor de presión del servofreno
- P0556 Rango incorrecto en el sensor de presión del servofreno
- P0557 Señal baja en el sensor de presión del servofreno
- P0558 Señal alta en el sensor de presión del servofreno

- P0559 Señal intermitente en el sensor de presión del servofreno
- P0560 Tensión del sistema incorrecta
- P0561 Tensión del sistema inestable
- P0562 Tensión del sistema baja
- P0563 Tensión del sistema Alta
- P0564 Circuito defectuoso en la señal de entrada del control de crucero.
- P0565 Encendido del control cruise defectuoso. Circuito
- P0566 Interruptor de apagado control cruise defectuoso
- P0567 Interruptor Resume o reanudación defectuoso
- P0568 Interruptor SET del control de crucero defectuoso
- P0569 Interruptor reducción de la velocidad del control de crucero defectuoso
- P0570 Interruptor aumento de la velocidad del control de crucero defectuoso
- P0571 Interruptor de velocidad/freno 1 defectuoso
- P0572 Señal baja interruptor de velocidad/freno
- P0573 Señal alta interruptor de velocidad /freno
- P0574 Control de crucero-Velocidad del vehículo muy alta
- P0575 Control velocidad de crucero. Señal defectuosa
- P0576 Señal baja control de crucero
- P0577 Señal alta control de crucero.
- P0578 Activación permanente del control de crucero. Circuito 1
- P0579 Rango incorrecto del control de velocidad del crucero Circuito 1
- P0580 Señal baja control de velocidad de crucero Circuito 1
- P0581 Señal alta control de velocidad de crucero circuito 1
- P0582 Control de velocidad de crucero control de vacío circuito abierto
- P0583 Señal baja del control de vacío del control de crucero
- P0584 Señal alta del control de vacío del control de crucero
- P0585 Correlación mal entre 1/2 control de velocidad de crucero
- P0586 Control de ventilación del control de crucero con el circuito abierto
- P0587 Señal baja del control de ventilación del control de crucero
- P0588 Señal alta del control de ventilación del control de crucero
- P0589 Circuito defectuoso control de velocidad de crucero circuito 2
- P0590 Activación permanente control de velocidad de crucero circuito 2
- P0591 Rendimiento incorrecto control de velocidad de crucero circuito 2
- P0592 Señal baja control de velocidad de crucero circuito 2
- P0593 Señal alta control de velocidad de crucero circuito 2
- P0594 Circuito abierto control del actuador del control de crucero
- P0595 Señal baja control del actuador de velocidad de crucero
- P0596 Señal alta control de velocidad de crucero control del actuador.
- P0597 Circuito abierto en el control del calentador del termostato
- P0598 Señal baja control calentador del termostato
- P0599 Señal alta control calentador del termostato
- P0600 Sistema Canbus defectuoso.
- P0601 Fallo en la memoria ROM Modulo control del motor

- P0602 Error de programación modulo control del motor
- P0603 Error permanente en memoria KAM módulo de control del motor.
- P0604 Error memoria Ram-Modulo control motor
- P0605 Error memoria ROM en el módulo control motor
- P0606 Fallo del procesador-Modulo control del motor
- P0607 Problema de funcionamiento modulo control de motor
- P0608 Señal de salida 1 defectuosa en el sensor de velocidad del vehículo
- P0609 Señal de salida 2 defectuosa en el módulo control del motor
- P0610 Error en las opciones del vehículo-Modulo de control
- P0611 Problema de funcionamiento en el módulo de control de inyección de combustible
- P0612 Fallo en el relé del módulo de control de combustible
- P0613 Fallo en el procesador del módulo de la caja de cambios. TCM
- P0614 ECM / TCM Incompatible
- P0615 Circuito defectuoso en el relé del motor de arranque
- P0616 Señal baja en el relé del motor de arranque
- P0617 Señal alta en el relé del motor de arranque
- P0618 Error de memoria permanente en el módulo de control de combustible alternativo
- P0619 Error de memoria RAM/ROM en el módulo de control de combustible alternativo
- P0620 Circuito defectuoso en el control del alternador
- P0621 Circuito defectuoso del testigo del alternador
- P0622 Circuito defectuoso en el testigo del alternador, control de campo.
- P0623 Circuito defectuoso testigo control alternador
- P0624 Circuito defectuoso en el testigo del tapón de llenado.
- P0625 Señal baja en el terminal campo alternador
- P0626 Señal alta en el terminal campo alternador
- P0627 Circuito abierto en el control de la bomba de combustible 1
- P0628 Señal baja en el control de la bomba de combustible 1
- P0629 Señal alta en el control de la bomba de combustible
- P0630 Número de bastidor no corresponde con el modulo motor/transmisión
- P0631 Número de bastidor incompatible con la caja de cambios TCM
- P0632 Cuentakilómetros No Programado – ECM / PCM
- P0633 Llave no programada modulo motor/transmisión
- P0634 PCM / ECM / TCM temperatura interna muy alta
- P0635 Circuito defectuoso en el control de la dirección asistida.
- P0636 Señal baja en el control de la dirección asistida
- P0637 Señal alta en el control de la dirección asistida
- P0638 Funcionamiento incorrecto del módulo de control de la mariposa bloque 1
- P0639 Funcionamiento incorrecto del módulo de control de la mariposa bloque 2
- P0640 Circuito defectuoso control de calentador de aire de la admisión.
- P0641 Circuito abierto tensión referencia sensor 1

- P0642 Señal baja tensión de referencia sensor 1
- P0643 Señal alta tensión de referencia sensor 1
- P0644 Circuito defectuoso comunicación en serie pantalla del conductor
- P0645 Funcionamiento incorrecto relé embrague compresor aire acondicionado.
- P0646 Señal baja del relé del embrague compresor aire acondicionado.
- P0647 Señal alta del relé del embrague compresor aire acondicionado.
- P0648 Circuito defectuoso testigo control inmovilizador
- P0649 Circuito defectuoso testigo velocidad de crucero.
- P0650 Circuito defectuoso testigo de avería
- P0651 Circuito abierto tensión de referencia 2
- P0652 Señal baja tensión de referencia 2
- P0653 Señal alta tensión de referencia 2
- P0654 Circuito defectuoso señal de salida régimen motor.
- P0655 Circuito defectuoso testigo sobrecalentamiento motor
- P0656 Circuito defectuoso señal salida de combustible.
- P0657 Circuito abierto tensión alimentación actuador 1
- P0658 Señal baja tensión alimentación actuador 1
- P0659 Señal alta tensión alimentación actuador 1
- P0660 Circuito abierto válvula control aire colector de admisión bloque 1
- P0661 Señal baja válvula control aire colector de admisión bloque 1
- P0662 Señal alta válvula control aire colector de admisión bloque 1
- P0663 Circuito abierto válvula control aire colector de admisión bloque 2
- P0664 Señal baja válvula control aire colector de admisión bloque 2
- P0665 Señal alta válvula control aire colector de admisión bloque 2
- P0666 Circuito defectuoso sensor temperatura interna modulo transmisión/motor
- P0667 Rendimiento incorrecto sensor temperatura interna modulo motor/transmisión
- P0668 Señal baja sensor temperatura interna modulo motor/transmisión
- P0669 Señal alta sensor temperatura interna modulo motor/transmisión
- P0670 Circuito defectuoso modulo control calentadores
- P0671 Circuito defectuoso calentadores cilindro 1
- P0672 Circuito defectuoso calentadores cilindro 2
- P0673 Circuito defectuoso calentadores cilindro 3
- P0674 Circuito defectuoso calentadores cilindro 4
- P0675 Circuito defectuoso calentadores cilindro 5
- P0676 Circuito defectuoso calentadores cilindro 6
- P0677 Circuito defectuoso calentadores cilindro 7
- P0678 Circuito defectuoso calentadores cilindro 8
- P0679 Circuito defectuoso calentadores cilindro 9
- P0680 Circuito defectuoso calentadores cilindro 10
- P0681 Circuito defectuoso calentadores cilindro 11
- P0682 Circuito defectuoso calentadores cilindro 12
- P0683 Comunicación incorrecta modulo calentadores/motor/transmisión

- P0684 Funcionamiento incorrecto en la comunicación entre los módulos de calentadores/motor /transmisión
- P0685 Circuito abierto relé alimentación modulo motor/transmisión
- P0686 Señal baja relé alimentación modulo motor/transmisión
- P0687 Señal alta relé alimentación modulo motor/transmisión
- P0688 Cortocircuito a positivo relé control del motor.
- P0689 Señal baja relé alimentación modulo motor/transmisión
- P0690 Señal alta relé alimentación modulo motor/transmisión.
- P0691 Cortocircuito a masa ventilador1 refrigerante motor
- P0692 Cortocircuito a positivo ventilador1 refrigerante motor
- P0693 Cortocircuito a masa ventilador 2 refrigerante motor
- P0694 Cortocircuito a positivo ventilador 2 refrigerante motor
- P0695 Cortocircuito a masa ventilador 3 refrigerante motor
- P0696 Cortocircuito a positivo ventilador 3 refrigerante motor
- P0697 Circuito abierto tensión referencia 3
- P0698 Señal baja sensor tensión referencia 3
- P0699 Señal alta sensor tensión referencia 3
- P0700 Sistema de control de transmisión defectuoso.
- P0701 Rendimiento incorrecto sistema de control de transmisión
- P0702 Fallo eléctrico sistema de control de transmisión
- P0703 Circuito defectuoso convertidor de par(interruptor de freno 2)
- P0704 Circuito defectuoso interruptor posición pedal embrague.
- P0705 Circuito defectuoso sensor /interruptor marchas cortas y largas L/D/N/R/P
- P0706 Rendimiento incorrecto sensor /interruptor marchas cortas y largas L/D/N/R/P
- P0707 Señal baja sensor /interruptor marchas cortas y largas L/D/N/R/P
- P0708 Señal alta sensor /interruptor marchas cortas y largas L/D/N/R/P
- P0709 Circuito intermitente sensor /interruptor marchas cortas y largas L/D/N/R/P
- P0710 Circuito defectuoso sensor temperatura 1 del aceite transmisión.
- P0711 Rango incorrecto sensor temperatura 1 del aceite transmisión.
- P0712 Señal baja sensor temperatura 1 del aceite transmisión.
- P0713 Señal alta sensor temperatura 1 del aceite transmisión.
- P0714 Señal intermitente sensor temperatura 1 del aceite transmisión.
- P0715 Circuito defectuoso sensor velocidad transmisión 1
- P0716 Rango incorrecto sensor velocidad transmisión 1
- P0717 No hay señal sensor velocidad transmisión 1
- P0718 Señal intermitente sensor velocidad transmisión 1
- P0719 Señal baja interruptor de freno 2/convertidor de par
- P0720 Circuito defectuoso sensor velocidad del vehículo
- P0721 Rendimiento incorrecto sensor velocidad del vehículo
- P0722 No hay señal sensor velocidad del vehículo

- P0723 Velocidad intermitente sensor velocidad del vehículo
- P0724 Señal alta convertidor de par/interruptor de freno 2
- P0725 Circuito defectuoso señal entrada régimen motor
- P0726 Rendimiento incorrecto señal entrada régimen motor
- P0727 No existe señal de entrada de régimen motor
- P0728 Señal intermitente entrada régimen motor
- P0729 Marcha 6 = relación incorrecta
- P0730 Relación de marchas incorrecta
- P0731 Marcha 1 = relación incorrecta
- P0732 Marcha 2 = relación incorrecta
- P0733 Marcha 3 = relación incorrecta
- P0734 Marcha 4 = relación incorrecta
- P0735 Marcha 5 = relación incorrecta
- P0736 Marcha atrás = relación incorrecta
- P0737 Señal de salida incorrecto módulo de motor control de transmisión
- P0738 Señal baja módulo de motor control de transmisión
- P0739 Señal alta módulo de motor control de transmisión
- P0740 Circuito defectuoso válvula embrague del convertidor de par
- P0741 Funcionamiento incorrecto/desactivado válvula embrague del convertidor de par
- P0742 Activado permanente válvula embrague del convertidor de par
- P0743 Fallo en el circuito eléctrico válvula embrague del convertidor de par
- P0744 Fallo intermitente válvula embrague del convertidor de par
- P0745 Circuito defectuoso solenoide presión de a
- P0746 Solenoide presión aceite transmisión – funcionamiento, desactivado
- P0747 Solenoide presión aceite transmisión – activado permanente
- P0748 Solenoide presión aceite transmisión – circuito eléctrico
- P0749 Solenoide presión aceite transmisión – interrupción intermitente
- P0750 Electroválvula cambio A – circuito defectuoso
- P0751 Electroválvula cambio A – funcionamiento, desactivado
- P0752 Electroválvula cambio A – activado permanente
- P0753 Electroválvula cambio A – circuito eléctrico
- P0754 Electroválvula cambio A – interrupción intermitente
- P0755 Electrovalvula cambio B – circuito defectuoso
- P0756 Electrovalvula cambio B – funcionamiento,desactivado
- P0757 Electrovalvula cambio B – activado permanente
- P0758 Electrovalvula cambio B – circuito electrico
- P0759 Electrovalvula cambio B – interrupcion intermitente
- P0760 Electrovalvula cambio C – circuito defectuoso
- P0761 Electrovalvula cambio C – funcionamiento,desactivado
- P0762 Electrovalvula cambio C – activado permanente
- P0763 Electrovalvula cambio C – circuito electrico
- P0764 Electrovalvula cambio C – interrupcion intermitente

- P0765 Electrovalvula cambio D – circuito defectuoso
- P0766 Electrovalvula cambio D – funcionamiento,desactivado
- P0767 Electrovalvula cambio D – activado permanente
- P0768 Electrovalvula cambio D – circuito electrico
- P0769 Electrovalvula cambio D – interrupcion intermitente
- P0770 Electrovalvula cambio E – circuito defectuoso
- P0771 Electrovalvula cambio E – funcionamiento, desactivado
- P0772 Electrovalvula cambio E – activado permanente
- P0773 Electrovalvula cambio E – circuito electrico
- P0774 Electrovalvula cambio E – interrupcion intermitente
- P0775 Solenoide control presion B – circuito defectuoso
- P0776 Solenoide control presion B – funcionamiento, desactivado
- P0777 Solenoide control presion B – activado permanente
- P0778 Solenoide control presion B – averia electrica
- P0779 Solenoide control presion B – interrupcion intermitente
- P0780 Seleccion de marchas – cambio defectuoso
- P0781 Seleccion de marchas 1-2 – cambio defectuoso
- P0782 Seleccion de marchas 2-3 – cambio defectuoso
- P0783 Seleccion de marchas 3-4 – cambio defectuoso
- P0784 Seleccion de marchas 4-5 – cambio defectuoso
- P0785 Electrovalvula cambio/reglaje – circuito defectuoso
- P0786 Electrovalvula cambio/reglaje – rango,funcionamiento
- P0787 Electrovalvula cambio/reglaje – señal baja
- P0788 Electrovalvula cambio/reglaje – señal alta
- P0789 Electrovalvula cambio/reglaje – señal intermitente
- P0790 Interruptor seleccion modo cambio – circuito defectuoso
- P0791 Sensor velocidad giro arbol intermedio – circuito defectuoso
- P0792 Sensor velocidad giro arbol intermedio – rango,funcionamiento
- P0793 Sensor velocidad giro arbol intermedio – No hay señal
- P0794 Sensor velocidad giro arbol intermedio – averia intermitente
- P0795 Solenoide presion aceite transmision C – circuito defectuoso
- P0796 Solenoide presion aceite transmision C – funcionamiento,desactivado
- P0797 Solenoide presion aceite transmision C – activado permanente
- P0798 Solenoide presion aceite transmision C – averia electrica
- P0799 Solenoide presion aceite transmision C – averia intermitente
- P0800 Control caja transferencia, peticion testigo averias – funcionamiento incorrecto
- P0901 Actuador embrague – rango,funcionamiento
- P0902 Actuador embrague – señal baja
- P0903 Actuador embrague – señal alta
- P0904 Posicion neutral caja cambios – circuito defectuoso
- P0905 Posicion neutral caja cambios – rango, funcionamiento
- P0906 Posicion neutral caja cambios – señal baja

- P0907 Posicion neutral caja cambios – señal alta
- P0908 Posicion neutral caja cambios – interrupcion intermitente
- P0909 Error regulacion posicion neutral caja cambios
- P0910 Actuador seleccion posicion neutral caja cambios – circuito defectuoso
- P0911 Actuador seleccion posicion neutral caja cambios – rango,funcionamiento
- P0912 Actuador seleccion posicion neutral caja cambios – señal baja
- P0913 Actuador seleccion posicion neutral caja cambios – señal alta
- P0914 Circuito posicion cambio marchas – circuito defectuoso
- P0915 Circuito posicion cambio marchas – rango,funcionamiento
- P0916 Circuito posicion cambio marchas – señal baja
- P0917 Circuito posicion cambio marchas – señal alta
- P0918 Circuito posicion cambio marchas – interrupcion intermitente
- P0919 Control posicion cambio marchas – error
- P0920 Actuador avance cambio marchas – circuito defectuoso
- P0921 Actuador avance cambio marchas – rango, funcionamiento
- P0922 Actuador avance cambio marchas – señal baja
- P0923 Actuador avance cambio marchas – señal alta
- P0924 Actuador retroceso cambio marchas – circuito defectuoso
- P0925 Actuador retroceso cambio marchas – rango,funcionamiento
- P0926 Actuador retroceso cambio marchas – señal baja
- P0927 Actuador retroceso cambio marchas – señal alta
- P0928 Electrovalvula bloqueo cambio marchas – circuito defectuoso
- P0929 Electrovalvula bloqueo cambio marchas – rango,funcionamiento
- P0930 Electrovalvula bloqueo cambio marchas – señal baja
- P0931 Electrovalvula bloqueo cambio marchas – señal alta
- P0932 Sensor presion hidraulica – circuito defectuoso
- P0933 Sensor presion hidraulica – rango,funcionamiento
- P0934 Sensor presion hidraulica – señal entrada baja
- P0935 Sensor presion hidraulica – señal entrada alta
- P0936 Sensor presion hidraulica – interrupcion intermitente
- P0937 Sensor temperatura aceite hidraulico – circuito defectuoso
- P0938 Sensor temperatura aceite hidraulico – rango,funcionamiento
- P0939 Sensor temperatura aceite hidraulico – señal entrada baja
- P0940 Sensor temperatura aceite hidraulico – señal entrada alta
- P0941 Sensor temperatura aceite hidraulico – interrupcion intermitente
- P0942 Unidad presion hidraulica
- P0943 Unidad presion hidraulica – recorrido corto
- P0944 Unidad presion hidraulica – perdida presion
- P0945 Rele bomba hidraulica – circuito abierto
- P0946 Rele bomba hidraulica – rango,funcionamiento
- P0947 Rele bomba hidraulica – señal baja
- P0948 Rele bomba hidraulica – señal alta
- P0949 Cambio manual automatizado (ASM) – valores adaptativos no registrados

- P0950 Cambio manual automatizado (ASM) – circuito defectuoso
- P0951 Cambio manual automatizado (ASM) – rango,funcionamiento
- P0952 Cambio manual automatizado (ASM) – señal baja
- P0953 Cambio manual automatizado (ASM) – señal alta
- P0954 Cambio manual automatizado (ASM) – interrupcion intermitente
- P0955 Circuito modo cambio manual automatizado (ASM) – circuito defectuoso
- P0956 Circuito modo cambio manual automatizado (ASM) – rango,funcionamiento
- P0957 Circuito modo cambio manual automatizado (ASM) – señal baja
- P0958 Circuito modo cambio manual automatizado (ASM) – señal alta
- P0959 Circuito modo cambio manual automatizado (ASM) – interrupcion intermitente
- P0960 Solenoide control presion A – circuito defectuoso
- P0961 Solenoide control presion A – rango,funcionamiento
- P0962 Solenoide control presion A – señal baja
- P0963 Solenoide control presion A – señal alta
- P0964 Solenoide control presion B – circuito defectuoso
- P0965 Solenoide control presion B – rango,funcionamiento
- P0966 Solenoide control presion B – señal baja
- P0967 Solenoide control presion B – señal alta
- P0968 Solenoide control presion C – circuito defectuoso
- P0969 Solenoide control presion C – rango,funcionamiento
- P0970 Solenoide control presion C – señal baja
- P0971 Solenoide control presion C – señal alta
- P0972 Electrovalvula de cambio A – rango,funcionamiento
- P0973 Electrovalvula de cambio A – señal baja
- P0974 Electrovalvula de cambio A – señal alta
- P0975 Electrovalvula de cambio B – rango,funcionamiento
- P0976 Electrovalvula de cambio B – señal baja
- P0977 Electrovalvula de cambio B – señal alta
- P0978 Electrovalvula de cambio C – rango,funcionamiento
- P0979 Electrovalvula de cambio C – señal baja
- P0980 Electrovalvula de cambio C – señal alta
- P0981 Electrovalvula de cambio D – rango,funcionamiento
- P0982 Electrovalvula de cambio D – señal baja
- P0983 Electrovalvula de cambio D – señal alta
- P0984 Electrovalvula de cambio E – rango,funcionamiento
- P0985 Electrovalvula de cambio E – señal baja
- P0986 Electrovalvula de cambio E – señal alta
- P0987 Sensor/Interruptor presion aceite transmision E – circuito defectuoso
- P0988 Sensor/Interruptor presion aceite transmision E – rango,funcionamiento
- P0989 Sensor/Interruptor presion aceite transmision E – señal baja
- P0990 Sensor/Interruptor presion aceite transmision E – señal alta
- P0991 Sensor/Interruptor presion aceite transmision E – interrupcion intermitente

- P0992 Sensor/Interruptor presion aceite transmision F – circuito defectuoso
- P0993 Sensor/Interruptor presion aceite transmision F – rango,funcionamiento
- P0994 Sensor/Interruptor presion aceite transmision F – señal baja
- P0995 Sensor/Interruptor presion aceite transmision F – señal alta
- P0996 Sensor/Interruptor presion aceite transmision F – interrupcion intermitente
- P0997 Electrovalvula cambio F – rango,funcionamiento
- P0998 Electrovalvula cambio F – señal baja
- P0999 Electrovalvula cambio F – señal alta
- P1000 Prueba preparacion sistema no completada
- P1001 Diagnosis del sistema incompleta
- P1100 Sensor flujo masa aire – interrupcion intermitente
- P1101 Sensor flujo masa aire – fuera limites
- P1102 Sensor MAF menor que previsto
- P1103 Sensor MAF mayor que previsto
- P1104 Sensor MAF – circuito defectuoso
- P1105 Sensor presion barometrica – circuito defectuoso
- P1106 Alternador/Sensor MAP señal tension baja
- P1107 Alternador/Sensor MAP señal tension baja
- P1108 Testigo Bateria, alternador – circuito defectuoso
- P1109 Temperatura aire admision – averia intermitente
- P1110 Temperatura aire admision – circuito abierto
- P1111 Temperatura aire admision – interrupcion intermitente/alta
- P1112 Temperatura aire admision – interrupcion intermitente/baja
- P1113 Temperatura aire admision – circuito abierto
- P1114 Temperatura motor intermitente/Aire admision señal entrada baja
- P1115 Temperatura motor intermitente/Aire admision señal entrada alta
- P1116 Temperatura motor fuera de rango
- P1117 Sensor temperatura refrigerante motor – interrupcion intermitente
- P1118 Sensor Temperatura absoluta escape – señal entrada baja
- P1119 Sensor Temperatura absoluta escape – señal entrada alta
- P1120 Sensor posicion mariposa – fuera de rango
- P1121 Sensor posicion mariposa – intermitente,señal alta
- P1122 Sensor posicion mariposa – intermitente,señal baja
- P1123 Sensor posicion mariposa – valor elevado
- P1124 Sensor posicion mariposa – fuera limites
- P1125 Sensor posicion mariposa – interrupcion intermitente
- P1126 Sensor posicion mariposa – circuito defectuoso
- P1127 Temperatura escape – fuera de rango
- P1128 Sondas lambda 1 – transpuestos
- P1129 Sondas lambda 2 – transpuestos
- P1130 Sonda lambda – regulacion inyeccion limite
- P1131 Sonda lambda – mezcla pobre
- P1132 Sonda lambda – mezcla rica

- P1133 Sonda lambda 1 – Insuficiente
- P1134 Sonda lambda 1 – tiempo reaccion
- P1135 Sensor posicion pedal A – circuito defectuoso
- P1136 Ventilador motor – circuito defectuoso
- P1137 Sonda lambda 1 bloque 2 – mezcla pobre
- P1138 Sonda lambda 1 bloque 2 – mezcla rica
- P1139 Indicador agua en combustible – circuito defectuoso
- P1140 Agua en el combustible
- P1141 Indicador restriccion combustible – circuito defectuoso
- P1142 Restriccion combustible
- P1143 Valvula control aire asistido – rango,funcionamiento
- P1144 Valvula control aire asistido – circuito defectuoso
- P1145 Error de par calculado
- P1150 Sonda lambda – regulacion inyeccion al limite
- P1151 Sonda lambda – mezcla pobre
- P1152 Sonda lambda – mezcla rica
- P1153 Control combustible (Bloque 2) – mezcla pobre
- P1154 Control combustible (bloque 2) – mezcla rica
- P1155 Controlador alternativo combustible
- P1156 Interruptor seleccion combustible – circuito defectuoso
- P1157 Sonda lambda 2 bloque 2 – mezcla pobre
- P1158 Sonda lambda 2 bloque 2 – mezcla rica
- P1159 Motor paso a paso combustible – circuito defectuoso
- P1167 Fallo diagnosis, Mariposa no soltada
- P1168 Sensor combustible rail – valor muy bajo
- P1169 Sensor combustible rail – valor muy alto
- P1170 Solenoide desconexion motor – fallo
- P1171 Sensor rotor – fallo
- P1172 Control rotor – fallo
- P1173 Calibracion rotor – fallo
- P1174 Sensor arbol levas – fallo
- P1175 Control arbol levas – fallo
- P1176 Calibracion arbol levas – fallo
- P1177 Fallo sincronizacion
- P1178 Circuito abierto
- P1180 Sensor temperatura combustible -señal baja
- P1181 Sensor temperatura combustible -señal alta
- P1182 Solenoide desconexion combustible – circuito defectuoso
- P1183 Temperatura aceite motor – circuito defectuoso
- P1184 Temperatura aceite motor – fuera de rango
- P1185 Sensor temperatura bomba combustible – alto
- P1186 Sensor temperatura bomba combustible – bajo
- P1187 Seleccion variante

- P1188 Fallo calibracion memoria
- P1189 Velocidad Bomba – fallo señal
- P1190 Resistencia Calibracion fuera de rango
- P1191 Tension externa
- P1193 Controlador sobrecorriente EGR
- P1194 UCE conversor A/D
- P1195 Bomba combustible – fallo inicializar
- P1196 Tension apagado llave – valor alto
- P1197 Tension apagado llave – valor bajo
- P1198 Control rotor combustible – bajo combustible
- P1199 Nivel combustible – señal entrada baja
- P1200 Inyector – rango,funcionamiento
- P1200 Inyector – rango, funcionamiento
- P1201 Inyector Cilindro 1 – circuito abierto, cortocircuito
- P1202 Inyector Cilindro 2 – circuito abierto, cortocicuito
- P1203 Inyector Cilindro 3 – circuito abierto, cortocicuito
- P1204 Inyector Cilindro 4 – circuito abierto, cortocicuito
- P1205 Inyector Cilindro 5 – circuito abierto, cortocicuito
- P1206 Inyector Cilindro 6 – circuito abierto, cortocicuito
- P1209 Presión control inyector – fallo sistema
- P1210 Presión control inyector – valor superior esperado
- P1211 Presión control inyector – alta/baja
- P1212 Presión control inyector – sin señal
- P1213 Inyector arranque – circuito defectuoso
- P1214 Sensor posición pedal B – interrupción intermitente
- P1215 Sensor posición pedal C – señal entrada baja
- P1216 Sensor posición pedal C – señal entrada alta
- P1217 Sensor posición pedal C – interrupción intermitente
- P1218 Control Inyección – alto
- P1219 Control Inyección – bajo
- P1220 Control Mariposa – circuito defectuoso
- P1221 Control Tracción – circuito defectuoso
- P1222 Control Tracción salida – circuito defectuoso
- P1223 Parada de emergencia redundante
- P1224 Sensor posición Mariposa B – valor fuera de rango
- P1225 Sensor elevación aguja inyector.
- P1226 Sensor posición regulador cantidad combustible – circuito defectuoso
- P1227 Solenoide reglaje inyección – valor Sobrepresión
- P1228 Solenoide reglaje inyección – valor bajo presión
- P1229 Controlador bomba intercooler – fallo
- P1230 Bomba combustible velocidad baja – circuito defectuoso
- P1231 Bomba combustible secundaria, alta velocidad – señal baja
- P1232 Velocidad bomba combustible principal – circuito defectuoso

- P1233 Modulo Controlador Bomba combustible desconexión línea
- P1234 Modulo Controlador Bomba combustible desconexión línea
- P1235 Control bomba combustible – fuera de rango
- P1236 Control bomba combustible – fuera de rango
- P1237 Bomba combustible secundaria – circuito defectuoso
- P1238 Bomba combustible secundaria – circuito defectuoso
- P1239 Fallo velocidad bomba combustible
- P1240 Sensor alimentación corriente
- P1241 Sensor alimentación corriente – señal entrada baja
- P1242 Sensor alimentación corriente – señal entrada alta
- P1243 Fallo segunda bomba combustible o masa
- P1244 Alternador – señal entrada alta
- P1245 Alternador – señal entrada baja
- P1246 Alternador
- P1247 Presión turbo baja
- P1248 Presión turbo no detectada
- P1249 Válvula reglaje inyección – funcionamiento
- P1250 Solenoide Bomba – circuito defectuoso
- P1251 Solenoide mezcla aire – circuito defectuoso
- P1252 Correlación pedal PDS1 y LPDS alta
- P1253 Correlación pedal PDS1 y LPDS baja
- P1254 Correlación pedal PDS2 y LPDS alta
- P1255 Correlación pedal PDS2 y LPDS baja
- P1256 Correlación pedal PDS1 y HPDS
- P1257 Correlación pedal PDS2 y HPDS
- P1258 Correlación pedal PDS1 y PDS2
- P1259 Error señal inmovilizador a modulo motor
- P1260 Robo detectado – vehículo inmovilizado
- P1261 Cilindro 1 alto a bajo – corto
- P1262 Cilindro 2 alto a bajo – corto
- P1263 Cilindro 3 alto a bajo – corto
- P1264 Cilindro 4 alto a bajo – corto
- P1265 Cilindro 5 alto a bajo – corto
- P1266 Cilindro 6 alto a bajo – corto
- P1267 Cilindro 7 alto a bajo – corto
- P1268 Cilindro 8 alto a bajo – corto
- P1269 Código inmovilizador no programado
- P1270 Régimen motor/velocidad vehículo máximos alcanzados
- P1271 Cilindro 1 alto a bajo – abierto
- P1272 Cilindro 2 alto a bajo – abierto
- P1273 Cilindro 3 alto a bajo – abierto
- P1274 Cilindro 4 alto a bajo – abierto
- P1275 Cilindro 5 alto a bajo – abierto

- P1276 Cilindro 6 alto a bajo – abierto
- P1277 Cilindro 7 alto a bajo – abierto
- P1278 Cilindro 8 alto a bajo – abierto
- P1280 Presión control inyección – demasiado bajo
- P1281 Presión control inyección – demasiado alto
- P1282 Presión control inyección – excesiva
- P1283 Regulador presión inyector – circuito defectuoso
- P1284 Fallo en presión control inyector
- P1285 Sensor temperatura culata – sobrecalentamiento
- P1286 Impulso combustible – menor de lo previsto
- P1287 Impulso combustible – mayor de lo previsto
- P1288 Sensor temperatura culata – fuera de limites
- P1289 Sensor temperatura culata – señal entrada alta
- P1290 Sensor temperatura culata – señal entrada baja
- P1291 Inyector bloque 1 corto a masa o positivo
- P1292 Inyector bloque 2 corto a masa o positivo
- P1293 Inyector bloque 1 abierto
- P1294 Inyector bloque 2 abierto
- P1295 Múltiples fallos bloque 1
- P1296 Múltiples fallos bloque 2
- P1297 Inyector cortado
- P1298 Fallo IDM
- P1299 Sensor temperatura culada – sistema protección activo
- P1300 Fallo calibración compresor
- P1301 Calibración compresor alta
- P1302 Calibración compresor baja
- P1303 Fallo calibración EGR
- P1304 Calibración EGR alta
- P1305 Calibración EGR baja
- P1306 Relé Kickdown forzado – fallo circuito
- P1307 Relé Kickdown mantenido – fallo circuito
- P1308 Aire acondicionado – fallo circuito
- P1309 Fallo chip supervisor detonación
- P1313 Fallo catalizador/detonación bloque 1
- P1314 Fallo catalizador/detonación bloque 2
- P1315 Detonación persistente
- P1316 Circuito inyector/Códigos IDM
- P1317 Circuito inyector/Códigos IDM
- P1319 Sensor posición pistón reglaje inyección – circuito defectuoso
- P1336 Sensor árbol levas – rango, funcionamiento
- P1340 Sensor posición árbol levas B – circuito defectuoso
- P1341 Sensor posición árbol levas B – rango, funcionamiento
- P1342 Sensor posición pedal acelerador A – rango, funcionamiento

- P1343 Sensor posición pedal acelerador B – rango, funcionamiento
- P1344 Sensor posición pedal acelerador C – rango, funcionamiento
- P1345 Sensor posición árbol levas – circuito defectuoso
- P1346 Sensor nivel combustible B – circuito defectuoso
- P1347 Sensor nivel combustible B – rango, funcionamiento
- P1348 Sensor nivel combustible B – señal baja
- P1349 Sensor nivel combustible B – señal alta
- P1350 Sensor nivel combustible B – interrupción intermitente
- P1351 Monitor diagnosis encendido – circuito entrada defectuoso
- P1352 Primario encendido A – circuito defectuoso
- P1353 Primario encendido B – circuito defectuoso
- P1354 Primario encendido C – circuito defectuoso
- P1355 Primario encendido D – circuito defectuoso
- P1358 Monitor diagnosis encendido – señal fuera de limites
- P1359 Sistema encendido, señal salida chispa – circuito defectuoso
- P1360 Secundario encendido A – circuito defectuoso
- P1361 Control encendido – tensión baja
- P1362 Secundario encendido C – circuito defectuoso
- P1363 Secundario encendido D – circuito defectuoso
- P1364 Primario encendido – circuito defectuoso
- P1365 Secundario encendido – circuito defectuoso
- P1366 Componentes encendido
- P1367 Componentes encendido
- P1368 Componentes encendido
- P1369 Testigo temperatura motor – circuito defectuoso
- P1370 Insuficiente incremento RPM
- P1371 Encendido cilindro 1 – fallo activación
- P1372 Encendido cilindro 2 – fallo activación
- P1373 Encendido cilindro 3 – fallo activación
- P1374 Encendido cilindro 4 – fallo activación
- P1375 Encendido cilindro 5 – fallo activación
- P1376 Encendido cilindro 6 – fallo activación
- P1380 Detectada detonación
- P1381 Actuador posición árbol levas (bloque 1) – encendido sobre avanzado
- P1382 Solenoide posición árbol levas (bloque 1) – circuito defectuoso
- P1383 Actuador posición árbol levas (bloque 1) – encendido sobre retrasado
- P1384 Solenoide A árbol levas – circuito defectuoso
- P1385 Solenoide B árbol levas – circuito defectuoso
- P1386 Actuador posición árbol levas (bloque 2) – encendido sobre avanzado
- P1387 Solenoide posición árbol levas (bloque 2) – circuito defectuoso
- P1388 Actuador posición árbol levas (bloque 2) – encendido sobre retrasado
- P1389 Bujías – señal baja
- P1390 Conector codificación octano – circuito abierto

- P1391 Bujías encendido (bloque 1) – señal baja
- P1392 Bujías encendido (bloque 1) – señal alta
- P1393 Bujías encendido (bloque 2) – señal baja
- P1394 Bujías encendido (bloque 2) – señal alta
- P1395 Fallo supervisor bujías (bloque 1)
- P1396 Fallo supervisor bujías (bloque 2)
- P1397 Tensión alimentación fuera de rango
- P1398 Solenoide B árbol levas – señal alta
- P1399 Bujías encendido – señal alta
- P1400 Sensor presión recirculación gases escape – señal baja
- P1401 Sensor presión recirculación gases escape – señal alta
- P1402 Sistema recirculación gases escape EGR – orificio medición restringido
- P1403 Sensor EGR – Diferencia control
- P1404 Temperatura aire inyección/Recirculación EGR
- P1405 Sensor EGR – tubo flexible delantero desconectado u obstruido
- P1406 Sensor EGR – tubo flexible trasero desconectado u obstruido
- P1407 Sensor EGR – No detecta flujo
- P1408 Recirculación gases escape – flujo fuera limites
- P1409 Solenoide recirculación gases escape – circuito defectuoso
- P1411 Inyección aire secundario – flujo bajo
- P1413 Supervisor inyección aire secundario – señal entrada baja
- P1414 Supervisor inyección aire secundario – señal entrada alta
- P1415 Bomba Aire – circuito defectuoso
- P1416 Entrada aire – circuito defectuoso
- P1417 Entrada aire – circuito defectuoso
- P1418 División aire 1 – circuito defectuoso
- P1419 División aire 2 – circuito defectuoso
- P1420 Sensor temperatura catalizador
- P1421 Catalizador – defectuoso
- P1422 Encendido gases escape – Sensor temperatura
- P1423 Encendido gases escape – prueba funcional
- P1424 Encendido gases escape – bujía primaria
- P1425 Encendido gases escape – bujía secundaria
- P1426 Encendido gases escape – señal MAF fuera de rango
- P1427 Encendido gases escape – señal MAF cortocircuito
- P1428 Encendido gases escape – señal MAF circuito abierto
- P1429 Bomba aire eléctrica – primaria
- P1430 Bomba aire eléctrica – secundaria
- P1432 Calentador termostato motor – circuito defectuoso
- P1433 Temperatura refrigerante aire acondicionado – señal baja
- P1434 Temperatura refrigerante aire acondicionado – señal alta
- P1435 Temperatura refrigerante aire acondicionado – rango, funcionamiento
- P1436 Temperatura evaporador aire acondicionado – señal baja

- P1437 Temperatura evaporador aire acondicionado – señal alta
- P1438 Temperatura evaporador aire acondicionado – rango, funcionamiento
- P1439 Interruptor temperatura – circuito defectuoso
- P1440 Válvula purga abierto
- P1441 Sistema emisión evaporaciones
- P1442 Control emisión evaporaciones
- P1443 Válvula control emisión evaporaciones
- P1444 Sensor purga flujo – señal baja
- P1445 Sensor purga flujo – señal alta
- P1446 Solenoide evaporaciones – circuito defectuoso
- P1447 Válvula ELC
- P1448 Fallo sistema ELC
- P1449 Solenoide chequeo evaporaciones – circuito defectuoso
- P1450 Deposito combustible
- P1451 Válvula control emisiones evaporación
- P1452 Deposito combustible
- P1453 Válvula presión deposito combustible
- P1454 Prueba sistema evaporaciones – circuito defectuoso
- P1455 Control emisión evaporaciones
- P1456 Sensor temperatura deposito combustible – circuito defectuoso
- P1457 Imposible forzar aspiración en depósito combustible
- P1460 Señal mariposa plena carga (corte A/C) – circuito defectuoso
- P1461 Sensor presión aire acondicionado – tensión baja
- P1462 Sensor presión aire acondicionado – tensión alta
- P1463 Sensor presión aire acondicionado – presión insuficiente
- P1464 Petición aire acondicionado fuera limites diagnosis
- P1465 Relé aire acondicionado – circuito defectuoso
- P1466 Sensor temperatura refrigerante aire acondicionado – circuito defectuoso
- P1467 Sensor temperatura compresor aire acondicionado – circuito defectuoso
- P1468 Circuito abierto/cortocircuito SSPOD
- P1469 Periodo reciclaje aire acondicionado bajo
- P1470 Embrague compresor aire acondicionado – recorrido demasiado corto
- P1471 Fallo electroventilador 1 (lado conductor)
- P1472 Fallo electroventilador 2 (lado pasajero)
- P1473 Ventilador secundario alto con ventiladores apagados
- P1474 Ventilador – circuito defectuoso
- P1475 Relé ventiladores bajo – circuito defectuoso
- P1476 Relé ventiladores alto – circuito defectuoso
- P1477 Relé ventilador adicional – circuito defectuoso
- P1478 Ventilador refrigerante motor – fallo controlador
- P1479 http://www.teseomotor.com
- P1480 Ventilador secundario bajo con ventiladores bajos encendidos
- P1481 Ventilador lento motor con ventilador rápido encendido

- P1482 Bomba combustible
- P1483 Alimentación ventiladores – sobre corriente
- P1484 Alimentación abierta a masa
- P1485 Calcula EGR – circuito defectuoso
- P1486 Actuador EGR – circuito defectuoso
- P1487 Solenoide EGR – circuito defectuoso
- P1490 Solenoide aire secundario – circuito defectuoso
- P1491 Interruptor solenoide secundario – circuito defectuoso
- P1492 Solenoide APLSOL – circuito defectuoso
- P1493 Solenoide RCNT – circuito defectuoso
- P1494 Solenoide SPCUT – circuito defectuoso
- P1495 Solenoide TCSPL – circuito defectuoso
- P1500 Sensor velocidad vehículo
- P1501 Sensor velocidad vehículo – fuera de limites
- P1502 Sensor velocidad vehículo – interrupción intermitente
- P1503 Sensor velocidad auxiliar
- P1504 Válvula control aire ralentí – circuito defectuoso
- P1505 Adaptación control aire ralentí
- P1506 Válvula control aire ralentí – error sobre velocidad
- P1507 Válvula control aire ralentí – error infravelocidad
- P1508 Solenoide subida ralentí 1 – circuito abierto
- P1509 Control ralentí – cortocircuito
- P1510 Señal ralentí – circuito defectuoso
- P1511 Interruptor ralentí, mariposa electrónica – circuito defectuoso
- P1512 Control aire colector admisión (bloque 1) – cerrado
- P1513 Control aire colector admisión (bloque 2) – cerrado
- P1514 Fallo controlador
- P1515 Corriente eléctrica – circuito defectuoso
- P1516 Solenoide aire colector admisión (bloque 1) – error señal entrada
- P1517 Solenoide aire colector admisión (bloque 2) – error señal entrada
- P1518 Control aire colector admisión – abierto
- P1519 Control aire colector admisión – cerrado
- P1520 Solenoide control aire colector admisión – circuito defectuoso
- P1521 Solenoide control aire colector admisión (bloque 1) – circuito defectuoso
- P1522 Solenoide control aire colector admisión (bloque 2) – circuito defectuoso
- P1523 Solenoide admisión variable – circuito defectuoso
- P1524 Solenoide admisión variable
- P1525 Válvula aire bypass
- P1526 Sistema aire bypass
- P1527 Solenoide acelerador – circuito defectuoso
- P1528 Solenoide válvula mariposa auxiliar – circuito defectuoso
- P1529 Solenoide SCAIR – circuito defectuoso
- P1530 Circuito aire acondicionado

- P1531 Movimiento pedal acelerador
- P1532 Circuito IMCC (Bloque B) defectuoso
- P1533 Circuito AAI defectuoso
- P1534 Interruptor inercia activado
- P1535 Velocidad ventilador – rango, funcionamiento
- P1536 Interruptor freno aparcamiento – circuito defectuoso
- P1537 Control aire colector admisión (bloque 1) – abierto
- P1538 Control aire colector admisión (bloque 2) – abierto
- P1539 Alimentación circuito aire acondicionado – sobre corriente
- P1540 Válvula aire bypass – circuito defectuoso
- P1549 Circuito IMCC (Bloque B) defectuoso
- P1550 PSPS fuera de rango
- P1563 Bomba inyección – petición parada motor
- P1564 Bomba inyección – petición modo combustible reducido
- P1565 Interruptor control velocidad – fuera de rango, alto
- P1566 Interruptor control velocidad – fuera de rango, bajo
- P1567 Salida control velocidad – continuidad
- P1568 Control velocidad – imposible mantener velocidad
- P1571 Interruptor frenos – circuito defectuoso
- P1572 Interruptor pedal freno – circuito defectuoso
- P1573 Posición mariposa no disponible
- P1574 Sensor posición mariposa – contradicción entre sensores
- P1575 Posición del pedal – fuera de rango
- P1576 Posición del pedal – no disponible
- P1577 Posición del pedal – contradicción entre sensores
- P1578 Alimentación ETC menor que exigida
- P1579 Alimentación ETC al limite
- P1580 Supervisor mariposa electrónica
- P1581 Supervisor mariposa electrónica – circuito defectuoso
- P1582 Supervisor mariposa electrónica – datos
- P1583 Supervisor mariposa electrónica – crucero desactivado
- P1584 Unidad Control mariposa Detecta IPE – circuito defectuoso
- P1585 Unidad control mariposa – circuito defectuoso
- P1586 Unidad control mariposa – posición mariposa defectuosa
- P1587 Unidad control mariposa modulada – circuito defectuoso
- P1588 Unidad control mariposa detecta perdida de retorno
- P1589 Unidad control mariposa no puede controlar Angulo mariposa deseado
- P1600 Perdida alimentación corriente auxiliar
- P1601 Error comunicaciones serie
- P1602 Modulo control inmovilizador – error comunicación
- P1603 Memoria EEPROM defectuosa
- P1604 Código no registrado
- P1605 Fallo memoria permanente

- P1606 Relé control O/P – circuito defectuoso
- P1607 Testigo Averías O/P – circuito defectuoso
- P1608 Señal control defectuosa
- P1609 Testigo averías – fallo controlador
- P1610 Códigos interactivos SBDS
- P1611 Códigos interactivos SBDS
- P1612 Códigos interactivos SBDS
- P1613 Códigos interactivos SBDS
- P1614 Códigos interactivos SBDS
- P1615 Códigos interactivos SBDS
- P1616 Códigos interactivos SBDS
- P1617 Códigos interactivos SBDS
- P1618 Códigos interactivos SBDS
- P1619 Códigos interactivos SBDS
- P1620 Códigos interactivos SBDS
- P1621 Memoria UCE/Código inmovilizador no coinciden
- P1622 Identificación inmovilizador no coincide
- P1623 Código inmovilizador/identificador – fallo escritura
- P1624 Sistema antideslizamiento
- P1625 Alimentación positivo a ventilador – circuito defectuoso
- P1626 Señal activado deslizamiento no recibida
- P1627 Modulo control motor – tensión alimentación fuera de rango
- P1628 Modulo control motor – tensión alimentación
- P1629 Modulo control motor – regulador interno tensión
- P1630 Modulo control motor – tensión referencia interna
- P1631 Relé control motor/Principal alimentación
- P1632 Sensor avería alternador – circuito defectuoso
- P1633 Tensión auxiliar – demasiado baja
- P1634 Transmisión salida datos – circuito defectuoso
- P1635 Valores fuera de rango
- P1636 Error comunicación chip señal inductiva
- P1637 Comunicación motor-ABS defectuosa
- P1638 Comunicación motor-cuadro instrumentos defectuoso
- P1639 Identificación vehículo errónea o no programada
- P1640 Extracción averías disponible en otro modulo
- P1641 Bomba combustible principal – circuito defectuoso
- P1642 Monitor bomba combustible – señal entrada alta
- P1643 Cableado circuito red módulos
- P1644 Control velocidad bomba combustible – circuito defectuoso
- P1645 Interruptor resistencia bomba combustible – circuito defectuoso
- P1650 Interruptor presión dirección asistida – fuera limites
- P1651 Interruptor presión dirección asistida – señal entrada
- P1652 Control Aire inyección desactivado

- P1653 Tensión Alimentación salida defectuosa
- P1654 Recirculación defectuosa
- P1655 Arranque desactivado – circuito defectuoso
- P1658 Bomba inyección – tensión alimentación fuera rango
- P1659 Bomba inyección – tensión alimentación
- P1660 Señal circuito salida alta
- P1661 Señal circuito salida baja
- P1662 Fallo circuito IDM_EN
- P1663 Señal petición combustible – circuito defectuoso
- P1664 Bomba inyección – funcionamiento incorrecto
- P1665 Bomba inyección – comunicación
- P1666 Bomba inyección – sincronización sensor posición cigüeñal
- P1667 Circuito Control Inyección defectuoso
- P1668 Bomba inyección – perdida señal comunicación
- P1669 Bomba inyección – supervisión
- P1670 Señal electrónica no detectada
- P1680 Medición bomba aceite – defectuosa
- P1681 Medición bomba aceite – defectuosa
- P1682 Medición bomba aceite – defectuosa
- P1683 Sensor temperatura bomba aceite – circuito defectuoso
- P1684 Sensor posición bomba aceite – circuito defectuoso
- P1685 Motor paso a paso bomba aceite – circuito defectuoso
- P1686 Motor paso a paso bomba aceite – circuito defectuoso
- P1687 Motor paso a paso bomba aceite – circuito defectuoso
- P1688 Motor paso a paso bomba aceite – circuito defectuoso
- P1689 Solenoide control presión aceite – circuito defectuoso
- P1690 Solenoide entrada – circuito defectuoso
- P1691 Solenoide control presión turbo – circuito defectuoso
- P1692 Solenoide control turbo – circuito defectuoso
- P1693 Control carga turbo – circuito defectuoso
- P1694 Carga turbo – circuito defectuoso
- P1695 Bus de datos CAN – datos bomba inyección
- P1700 Transmisión – fallo en posición punto muerto
- P1701 Error reversible
- P1702 Circuito Transmisión – interrupción intermitente
- P1703 Interruptor posición pedal freno – fuera de limites
- P1704 Fallo transición estados en Transmisión Digital
- P1705 Interruptor marchas cortas/largas – sin función en P/N
- P1706 Velocidad vehículo alta en aparcamiento
- P1707 Fallo indicador punto muerto
- P1708 Interruptor aire acondicionado – circuito defectuoso
- P1709 Interruptor posición estacionamiento/punto muerto – fuera limites
- P1711 sensor TFT fuera de rango

- P1712 Sistema reducción par transmisión – error señal
- P1713 Sensor TFT – Valor muy bajo
- P1714 Señal inductiva SSA defectuosa
- P1715 Señal inductiva SSB defectuosa
- P1716 Señal Inductiva SSC defectuosa
- P1717 Señal inductiva SSD defectuosa
- P1718 Sensor TFT – valor muy alto
- P1720 Medidor velocidad vehículo – circuito defectuoso
- P1721 Marcha 1 incorrecta
- P1722 Marcha 2 incorrecta
- P1723 Marcha 3 incorrecta
- P1724 Marcha 4 incorrecta
- P1725 Insuficiente velocidad motor incrementada durante diagnosis
- P1726 Insuficiente velocidad motor excrementada durante diagnosis
- P1727 Señal inductiva solenoide aire acondicionado – circuito defectuoso
- P1728 Error transmisión
- P1729 Error Interruptor 4×4 bajo
- P1730 Control marchas 2,3,5 – circuito defectuoso
- P1731 Cambio marchas 1-2 – circuito defectuoso
- P1732 Cambio marchas 2-3 – circuito defectuoso
- P1733 Cambio marchas 3-4 – circuito defectuoso
- P1734 Control marchas – circuito defectuoso
- P1735 Interruptor marcha primera – circuito defectuoso
- P1736 Interruptor marcha segunda – circuito defectuoso
- P1737 Solenoide bloqueo sistema
- P1738 Error tiempo cambio
- P1739 Solenoide sistema
- P1740 Señal inductiva convertidor embrague rotativo – circuito defectuoso
- P1741 Error control convertidor embrague rotativo
- P1742 Fallo Solenoide convertidor embrague rotativo
- P1743 Fallo Solenoide convertidor embrague rotativo
- P1744 Convertidor embrague rotativo – funcionamiento
- P1745 Solenoide presión sistema
- P1746 Solenoide control presión A – circuito abierto
- P1747 Solenoide control presión A – cortocircuito
- P1748 EPC – circuito defectuoso
- P1749 Solenoide control presión – fallo bajo
- P1751 Solenoide cambio A – funcionamiento
- P1754 Solenoide embrague – circuito defectuoso
- P1755 Sensor velocidad intermedio – circuito defectuoso
- P1756 Solenoide cambio B – funcionamiento
- P1760 Solenoide control presión A – cortocircuito
- P1761 Solenoide cambio C – funcionamiento

- P1762 Fallo Margen superior
- P1765 Solenoide reglaje inyección – circuito defectuoso
- P1767 Convertidor embrague rotativo – circuito defectuoso
- P1768 modo funcionamiento normal/invierno – circuito defectuoso
- P1769 Fallo modulación par transmisión (AG4)
- P1770 Solenoide embrague – circuito defectuoso
- P1775 Fallo testigo averías transmisión
- P1776 Fallo petición retardo encendido
- P1777 Fallo petición retardo encendido
- P1778 Transmisión I/P – circuito defectuoso
- P1779 Circuito TCIL defectuoso
- P1780 Interruptor control transmisión – fuera de rango
- P1781 Interruptor 4X4 – fuera de rango
- P1782 Circuito P/ES fuera de rango
- P1783 Condición sobre temperatura en transmisión
- P1784 Fallo mecánico en transmisión – Primera y Atrás
- P1785 Fallo mecánico en transmisión – Primera y Segunda
- P1786 Error cambio marchas 3-2
- P1787 Error cambio marchas 2-1
- P1788 Solenoide control presión B – circuito abierto
- P1789 Solenoide control presión B – cortocircuito
- P1790 Transmisión mecánica – circuito defectuoso
- P1791 Transmisión eléctrica – circuito defectuoso
- P1792 Sensor presión barométrica – circuito defectuoso
- P1793 Volumen aire admisión – circuito defectuoso
- P1794 Tensión batería
- P1795 Interruptor ralentí – circuito defectuoso
- P1796 Interruptor KickDown – circuito defectuoso
- P1797 Interruptor punto muerto – circuito defectuoso
- P1798 Temperatura refrigerante – circuito defectuoso
- P1799 Interruptor mantenido – circuito defectuoso
- P1800 Interruptor seguridad bloqueo embrague/transmisión – circuito defectuoso
- P1801 Interruptor seguridad bloqueo embrague/transmisión – circuito abierto
- P1802 Interruptor seguridad bloqueo embrague/transmisión – corto a positivo
- P1803 Interruptor seguridad bloqueo embrague/transmisión – corto a masa
- P1804 Indicador transmisión alta 4 ruedas – circuito defectuoso
- P1805 Indicador transmisión alta 4 ruedas – circuito abierto
- P1806 Indicador transmisión alta 4 ruedas – corto a positivo
- P1807 Indicador transmisión alta 4 ruedas – corto a masa
- P1808 Indicador transmisión baja 4 ruedas – circuito defectuoso
- P1809 Indicador transmisión baja 4 ruedas – circuito abierto
- P1810 Indicador transmisión baja 4 ruedas – corto a positivo
- P1811 Indicador transmisión baja 4 ruedas – corto a masa

- P1812 Modo selección transmisión 4 ruedas – circuito defectuoso
- P1813 Modo selección transmisión 4 ruedas – circuito abierto
- P1814 Modo selección transmisión 4 ruedas – corto a positivo
- P1815 Modo selección transmisión 4 ruedas – corto a masa
- P1816 Interruptor seguridad transmisión punto muerto – circuito defectuoso
- P1817 Interruptor seguridad transmisión punto muerto – circuito abierto
- P1818 Interruptor seguridad transmisión punto muerto – corto a positivo
- P1819 Interruptor seguridad transmisión punto muerto – corto a masa
- P1820 Relé cambio transferencia transmisión – circuito defectuoso
- P1821 Relé cambio transferencia transmisión – circuito abierto
- P1822 Relé cambio transferencia transmisión – corto a positivo
- P1823 Relé cambio transferencia transmisión – corto a masa
- P1824 Relé embrague transmisión 4 ruedas – circuito defectuoso
- P1825 Relé embrague transmisión 4 ruedas – circuito abierto
- P1826 Relé embrague transmisión baja 4 ruedas – corto a positivo
- P1827 Relé embrague transmisión baja 4 ruedas – corto a masa
- P1828 Relé cambio transferencia transmisión – circuito defectuoso
- P1829 Relé cambio transferencia transmisión – circuito abierto
- P1830 Relé cambio transferencia transmisión – corto a positivo
- P1831 Relé cambio transferencia transmisión – corto a masa
- P1832 Solenoide bloqueo diferencial transferencia transmisión – circuito defectuoso
- P1833 Solenoide bloqueo diferencial transferencia transmisión – circuito abierto
- P1834 Solenoide bloqueo diferencial transferencia transmisión – corto a positivo
- P1835 Solenoide bloqueo diferencial transferencia transmisión – corto a masa
- P1836 Sensor velocidad frontal transferencia transmisión – circuito defectuoso
- P1837 Sensor velocidad lateral transferencia transmisión – circuito defectuoso
- P1838 Motor cambio transferencia transmisión – circuito defectuoso
- P1839 Motor cambio transferencia transmisión – circuito abierto
- P1840 Motor cambio transferencia transmisión – corto a positivo
- P1841 Motor cambio transferencia transmisión – corto a masa
- P1842 Interruptor bloqueo diferencial transferencia transmisión – circuito defectuoso
- P1843 Interruptor bloqueo diferencial transferencia transmisión – circuito abierto
- P1844 Interruptor bloqueo diferencial transferencia transmisión – corto a positivo
- P1845 Interruptor bloqueo diferencial transferencia transmisión – corto a masa
- P1846 Contacto plata A transferencia transmisión – circuito defectuoso
- P1847 Contacto plata A transferencia transmisión – circuito abierto
- P1848 Contacto plata A transferencia transmisión. Transmisión – corto a positivo
- P1849 Contacto plata A transferencia transmision. Transmisión – corto a masa
- P1850 Contacto plata B transferencia transmisión. Transmisión – circuito defectuoso
- P1851 Contacto plata B transferencia transmisión. Transmisión – circuito abierto
- P1852 Contacto plata B transferencia transmisión – corto a positivo
- P1853 Contacto plata B transferencia transmisión – corto a masa

- P1854 Contacto plata C transferencia transmisión – circuito defectuoso
- P1855 Contacto plata C transferencia transmisión – circuito abierto
- P1856 Contacto plata C transferencia transmisión – corto a positivo
- P1857 Contacto plata C transferencia transmisión – corto a masa
- P1858 Contacto plata D transferencia transmisión – circuito defectuoso
- P1859 Contacto plata D transferencia transmisión – circuito abierto
- P1860 Contacto plata D transferencia transmisión – corto a positivo
- P1861 Contacto plata D transferencia transmisión – corto a masa
- P1862 Alimentación contacto plata transferencia transmisión – circuito defectuoso
- P1863 Alimentación contacto plata transferencia transmisión – circuito abierto
- P1864 Alimentación contacto plata transferencia transmisión – corto a positivo
- P1865 Alimentación contacto plata transferencia transmisión – corto a masa
- P1866 Transferencia transmisión – servicio requerido
- P1867 Contacto plata transferencia transmisión – circuito defectuoso
- P1868 Testigo transmisión automática 4 ruedas – circuito defectuoso
- P1869 Testigo transmisión automática 4 ruedas – corto a positivo
- P1870 Interruptor transferencia transmisión mecánica 4×4 – circuito defectuoso
- P1871 Interruptor transferencia transmisión mecánica 4×4 – corto a positivo
- P1872 Testigo bloqueo transmisión mecánica 4 ruedas – circuito defectuoso
- P1873 Testigo bloqueo transmisión mecánica 4 ruedas – corto a positivo
- P1874 Sensor alimentación transmisión automática efecto Hall – circuito defectuoso
- P1875 Sensor alimentación transmisión automática efecto Hall – corto a positivo
- P1876 Solenoide transferencia transmisión 2 ruedas – circuito defectuoso
- P1877 Solenoide transferencia transmisión 2 ruedas – corto a positivo
- P1878 Solenoide desencaje transferencia transmisión – circuito defectuoso
- P1879 Solenoide desencaje transferencia transmisión – circuito abierto
- P1880 Solenoide desencaje transferencia transmisión – corto a positivo
- P1881 Interruptor nivel refrigerante motor – circuito defectuoso
- P1882 Interruptor nivel refrigerante motor – corto a masa
- P1883 Interruptor nivel refrigerante motor – circuito defectuoso
- P1884 Testigo Interruptor nivel refrigerante motor – corto a masa
- P1885 Solenoide desencaje transferencia transmisión – corto a masa
- P1886 Fallo inicialización 4×4
- P1890 Selección modo transmisión 4WD – circuito defectuoso
- P1891 Contacto plata transferencia transmisión – circuito abierto
- P1900 Circuito OSS – interrupción intermitente
- P1901 Circuito TSS – Interrupción intermitente
- P1902 Solenoide B Control presión – intermitente/corto
- P1903 Solenoide C Control presión – cortocircuito
- P1904 Solenoide C Control presión – circuito abierto
- P1905 Solenoide C Control presión – intermitente/corto
- P1906 Relé kickdown forzado circuito abierto/cortocircuito masa
- P1907 Relé kickdown mantenido circuito abierto/cortocircuito masa

- P1908 Solenoide presión aceite transmisión – abierto/cortocircuito a masa
- P1909 Sensor temperatura aceite transmisión – abierto/cortocircuito a masa
- P1910 Fallo salida presión VFS A baja
- P1911 Fallo salida presión VFS B baja
- P1912 Fallo salida presión VFS C baja
- P1913 Interruptor A presión – circuito defectuoso
- P1914 Interruptor cambio Automático/Manual – circuito defectuoso
- P1915 Interruptor marcha atrás – circuito defectuoso
- P1916 Sensor velocidad altura cilindro embrague – circuito defectuoso
- P1917 Sensor velocidad altura cilindro embrague – interrupción intermitente
- P1918 Desplaye rango transmisión – circuito defectuoso
- P2000 Filtro óxidos nitrógeno

Los códigos de averías anteriores han sido extraídos de la siguiente web http://www.teseomotor.com/fallo-obd2/ , donde se detallan más ampliamente los tipos y causas de averías.

Bibliografía

[1] ARDUINO CC. HARDWARE. Web de Arduino. *Documentación necesaria para la descripción de hardware.* [consulta: 13/10/2015]. Disponible en: http: https://www.arduino.cc/en/Main/Products

[2] TEXAS INSTRUMENTS. Web de Texas Instruments. *Software and development tolos, LaunchPads.* [consulta: 15/10/2015]. Disponible en: http://www.ti.com/lsds/ti/tools-software/launchpads/launchpads.page

[3] SPARKFUN OBD-II UART. *Description.* [consulta: 14/10/2015]. Disponible en: https://www.sparkfun.com/products/9555

[4] TESEOMOTOR. Blog de Teseo Motor. *Trucos y consejos ELM327.* [consulta: 16/10/2015]. Disponible en: http://www.teseomotor.com/trucos-y-consejos-elm327/

[5] DESARROLLO DE APLICACIONES PARA ANDROID. Web FORMACAR. *Desarrollo de aplicaciones para Android.* [consulta: 16/02/2016]. Disponible en: https://www.formacarm.es/portal/servlet/formacarm.servlets.Portal?METHOD=DTEMARIO&id=69

[6] TOMÁS GIRONÉS, JESÚS. *El gran libro de Android.* Ed. [4ª ed.] Barcelona. Marcombo, Inc, 2015 [consulta: 02/02/2016]

[7] NAYLAMP MECHATRONICS. Blog de Naylamp mechatronics. *Tutorial Básico de Módulo Bluetooth HC-06 y HC-05.* [consulta: 18/03/2016]. Disponible en: http://www.naylampmechatronics.com/blog/12_Tutorial-B%C3%A1sico-de-Uso-del-M%C3%B3dulo-Bluetooth-H.html

[8] BLE SHIELD. Web de Kickstarter. *BLE SHIELD Description.* [consulta: 13/03/2016]. Disponible en: http://redbearlab.com/bleshield/

[9] PROMETEC. Web de Prometec. *Descripción del módulo Bluetooth ESp-8266 01 WIFI.* [consulta: 02/03/2016]. Disponible en: https://www.prometec.net/producto/esp-8266-01/

[10] ARDUINO CC. SOFTWARE. Foros de Arduino. *Librerías necesarias para la conexión de sensores.* [consulta: 13/10/2015]. Disponible en: http:// http://forum.arduino.cc/

[11] ELM327 v2.2. Web ELM Electonics. *ELM327 OBD to RS232 Interpreter-firmware versión 2.2.* [consulta: 11/11/2015]. Disponible en: https://www.elmelectronics.com/ic/elm327/

[12] Dra. NIEVES PAVÓN PULIDO. *Seminario de Android.* Departamento de Ingeniería de Sistemas y Automática, UPCT. M. en Comunicación Móvil y Contenido Digital, UMU. Curso 2016-2017(Segundo Cuatrimestre) [consulta: 05/07/2017]

[13] MEIER, Reto. *Professional Android 2 application development.* Ed. Indianapolis, IN: Wiley Pub., Inc., 2010. [consulta: 02/02/2016]

[14] INTRODUCCIÓN DE LA PROGRAMACIÓN EN JAVA. Web FORMACAR. [consulta: 21/10/2015]. Disponible en: https://www.formacarm.es/portal/servlet/formacarm.servlets.Portal?METHOD=DTEMARIO&id=66

[15] DESARROLLO DE APLICACIONES WEB CON JAVA. Web FORMACAR. [consulta: 24/05/2015]. Disponible en: https://www.formacarm.es/portal/servlet/formacarm.servlets.Portal?METHOD=DTEMARIO&id=75

Bibliografía

[16] APRENDIENDO A MANEJAR ARDUINO EN PROFUNDIDAD. Blog Aprendiendo Arduino. *Bluetooth en Arduino.* [consulta: 01/04/2016]. Disponible en: https://aprendiendoarduino.wordpress.com/tag/hc-05/

[17] LUIS LLAMAS INGENIERÍA, INFORMÁTICA Y DISEÑO. Blog Luis Llamas. *Conectar Arduino por Bluetooth con los módulos HC-05 o HC-06*. [consulta: 02/04/2016]. Disponible en: https://www.luisllamas.es/